AfIA

Association française
pour l'Intelligence Artificielle

CAp

Conférence sur l'Apprentissage automatique

PFIA 2019

Table des matières

Éditorial

La Conférence sur l'Apprentissage automatique (CAp) est le rendez-vous annuel de la communauté francophone dans le domaine de l'apprentissage automatique. Pour sa 21-ème édition, la conférence se tiendra du 3 au 5 juillet 2019 à Toulouse, ville caractérisée par la richesse de son potentiel de recherche académique et industrielle. La conférence est hébergée par la plateforme d'Intelligence Artificielle PFIA, au coté d'un grand nombre d'autres conférences et ateliers consacrés à l'IA. C'est ainsi l'occasion de mettre en avant l'ensemble des disciplines scientifiques et des domaines applicatifs à l'œuvre dans le développement de l'apprentissage automatique, telles l'informatique, les mathématiques appliquées, les sciences de l'ingénierie et des systèmes, les neurosciences et sciences cognitives, les sciences humaines et sociales. Comme pour les éditions précédentes, CAp 2019 est un lieu d'échanges et de convivialité pour tous les acteurs du domaine, avec cette année encore des conférenciers invités de renom.

La conférence CAp 2019 a lieu dans les locaux de l'Université Toulouse 1 Capitole. Elle est hébergée par la plateforme PFIA, avec le soutien de l'unité Inra MIAT, de la Société Savante Francophone d'Apprentissage Machine (SSFAM), ainsi que des sponsors qui ont entre autres rendu possible le tarif préférentiel accordé aux membres de la SSFAM.

Frédérick GARCIA

Comité de programme

Présidents

— Frédérick Garcia (INRA, MIAT, Toulouse)

Membres

— Massih-Reza Amini (Université Grenoble Alpes, LIG, Grenoble)
— Dominique Bontemps (Université Paul Sabatier, IMT, Toulouse)
— Stéphane Canu (Insa de Rouen)
— Marie-Josée Cros (INRA, MIAT, Toulouse)
— Nicolas Dobigeon (INP-ENSEEIHT, IRIT, Toulouse)
— Emmanuel Faure (CNRS, IRIT, Toulouse)
— Cédric Févotte (CNRS, IRIT, Toulouse)
— Sébastien Gerchinovitz (Université Paul Sabatier, IMT, Toulouse)
— Yves Grandvalet (CNRS, Heudiasyc, Compiègne)
— Timothée Masquelier (CNRS, CerCo, Toulouse)
— Thomas Oberlin (INP-ENSEEIHT, IRIT, Toulouse)
— Nathalie Peyrard (INRA, MIAT, Toulouse)
— Emmanuel Rachelson (ISAE, Toulouse)
— Régis Sabbadin (INRA, MIAT, Toulouse)
— Michèle Sebag (CNRS, LRI, Gif sur Yvette)
— Rufin Van Rullen (CNRS, CerCo, Toulouse)

Relecteurs

— Thierry Artières (Université d'Aix-Marseille, LIF)
— Stephane Ayache (Université d'Aix-Marseille, LIF)
— Nicolas Baskiotis (Sorbonne Université, LIP6)
— Aurélien Bellet (Inria)
— Younès Bennani (Université Paris13, LIPN)
— Marianne Clausel (Lorraine University)
— Stéphan Clémençon (Telecom ParisTech)
— Antoine Cornuéjols (AgroParisTech)
— Nicolas Courty (Université de Bretagne Sud, IRISA)
— Bruno Cremilleux (Université de Caen Normandie)
— Colin De La Higuera (University of Nantes)
— Gaël Dias (Normandie University)
— Remi Eyraud (Aix-Marseille University)
— Aurélien Garivier (Institut de Mathématiques de Toulouse)
— Gilles Gasso (INSA de Rouen, LITIS)
— Romaric Gaudel (Université de Rennes, Ensai, CREST)
— Pascal Germain (Inria)
— Remi Gilleron (Université de Lille, Inria)
— Alexandre Gramfort (Telecom ParisTech, CNRS LTCI)
— Yann Guermeur (CNRS)
— Vincent Guigue (Sorbonne Université, LIP6)
— Amaury Habrard (University of Saint-Etienne)
— Alexis Joly (Inria)
— Mikaela Keller (Université Lille 3, Inria)
— Christine Keribin (Université Paris Sud, Laboratoire de Mathématique)
— Engelbert Mephu Nguifo (University Clermont Auvergne, CNRS, LIMOS)
— Mohamed Nadif (University Paris Descartes)
— Liva Ralaivola (Université d'Aix-Marseille, LIF)
— Ievgen Redko (University Jean Monnet of Saint-Etienne, LHC)
— Marc Tommasi (University of Lille)

Context-Aware Zero-Shot Learning for Object Recognition

Éloi Zablocki *[1], Patrick Bordes *[1], Benjamin Piwowarski[1], Laure Soulier[1], et Patrick Gallinari[1,2]

[1]Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France
[2]Criteo AI Lab, Paris, France

May 26, 2019

Abstract

Zero-Shot Learning (ZSL) aims at classifying unlabeled objects by leveraging auxiliary knowledge, such as semantic representations. A limitation of previous approaches is that only intrinsic properties of objects, e.g. their visual appearance, are taken into account while their context, e.g. the surrounding objects in the image, is ignored. Following the intuitive principle that objects tend to be found in certain contexts but not others, we propose a new and challenging approach, *context-aware ZSL*, that leverages semantic representations in a new way to model the conditional likelihood of an object to appear in a given context. Finally, through extensive experiments conducted on Visual Genome, we show that contextual information can substantially improve the standard ZSL approach and is robust to unbalanced classes.

Keywords: zero-shot learning, visual context

1 Introduction

Traditional Computer Vision models, such as Convolutional Neural Networks (CNNs) [25], are designed to classify images into a set of predefined classes. Their performances have kept improving in the last decade, namely on object recognition benchmarks such as ImageNet [10], where state-of-the-art models [51, 37] have outmatched humans. However, training such models requires hundreds of manually-labeled instances for each class, which is a tedious and costly acquisition process. Moreover, these models cannot replicate humans' capacity to generalize and to recognize objects they have never seen before. As a response to these limitations, Zero-Shot Learning (ZSL) has emerged as an important research field in the last decade [11, 30, 15, 22]. In the object recognition field, ZSL aims at labeling an instance of a class for which no supervised data is available, by using knowledge acquired from another disjoint set of classes, for which corresponding visual instances are provided. In the literature, these sets of classes are respectively called *target* and *source* domains — terms borrowed from the transfer learning community. Generalization from the source to the target domain is achieved using auxiliary knowledge that semantically relates classes of both domains, e.g. attributes or textual representations of the class labels.

Previous ZSL approaches only focus on intrinsic properties of objects, e.g. their visual appearance, by the means of handcrafted features — e.g. shape, texture, or color — [24] or distributed representations learned from text corpora [1, 28]. The underlying hypothesis is that the identification of entities of the target domain is made possible thanks to the implicit *principle of compositionality* (a.k.a. Frege's principle) — an object is formed by the composition of its attributes and characteristics — and the fact that other entities of the source domain share the same attributes. For example, if textual resources state that an apple is round and that it can be red or green, this knowledge can be used to identify apples in images because these characteristics ('round', 'red', 'green') could be shared by classes of the source domain (e.g, 'round' like a tennis ball, 'red' like a strawberry...).

We believe that visual context, i.e. the other entities surrounding an object, also explains human's ability to recognize an object that has never been seen before. This assumption relies on the fact that scenes

*equal contribution

are *compositional* in the sense that they are formed by the composition of objects they contain. Some works in Computer Vision have exploited visual context to refine the predictions of classification [29] or detection [5] models. To the best of our knowledge, context has not been exploited in ZSL because, for obvious reasons, it is impossible to directly estimate the likelihood of a context for objects from the target domain — from visual data only. However, textual resources can be used to provide insights on the possible visual context in which an object is expected to appear. To illustrate this, knowing from language that an apple is likely to be found hanging on a tree or in the hand of someone eating it, can be very helpful to identify apples in images. In this paper, our goal is to leverage visual context as an additional source of knowledge for ZSL, by exploiting the distributed word representations [31] of the object class labels. More precisely, we adopt a probabilistic framework in which the probability to recognize a given object is split into three components: (1) a *visual component* based on its visual appearance (which can be derived from any traditional ZSL approach), (2) a *contextual component* exploiting its visual context, and (3) a *prior component*, which estimates the frequency of objects in the dataset. As a complementary contribution, we show that separating prior information in a dedicated component, along with simple yet effective sampling strategies, leads to a more interpretable model, able to deal with imbalanced datasets. Finally, as traditional ZSL datasets lack contextual information, we design a new dedicated setup based on the richly annotated Visual Genome dataset [23]. We conduct extensive experiments to thoroughly study the impact of contextual information.

2 Related work

Zero-shot learning While state-of-the-art image classification models [51, 37] restrict their predictions to a finite set of predefined classes, ZSL bypasses this important limitation by transferring knowledge acquired from seen classes (*source domain*) to unseen classes (*target domain)*. Generalization is made possible through the medium of a common semantic space where all classes from both source and target domains are represented by vectors called *semantic representations*.

Historically, the first semantic representations that were used were handcrafted attributes [11, 30, 24]. In these works, the attributes of a given image are determined and the class with the most similar attributes is predicted. Most methods represent class labels with binary vectors of visual features (e.g, 'IsBlack','HasClaws') [27, 14, 24]. However, attribute-based methods do not scale efficiently since the attribute ontology is often domain-specific and has to be built manually.

To cope with this limitation, more recent ZSL works rely on distributed semantic representations learned from textual datasets such as Wikipedia, using Distributional Semantic Models [31, 33, 34]. These models are based on the *distributional hypothesis* [19], which states that textual items with similar contexts in text corpora tend to have similar meanings. This is of particular interest in ZSL: all object classes (from both source and target domains) are embedded into the same continuous vector space based on their textual context, which is a rich source of semantic information. Some models directly aggregate textual representations of class labels and the predictions of a CNN [32], whereas others learn a cross-modal mapping between image representations (given by a CNN) and pre-learned semantic embeddings [2, 7]. At inference, the predicted class of a given image is the nearest neighbor in the semantic embedding space. The cross-modal mapping is linear in most of ZSL works [38, 1, 35]; this is the case in the present paper. Among these works, the DeViSE model [13] uses a max-margin ranking objective to learn a cross-modal projection and fine-tune the lower layers of the CNN. Several models have built upon DeViSE with approaches that learn non-linear mappings between the visual and textual modalities [3, 46], or by using a common multimodal space to embed both images and object classes [16, 28]. In this paper, we extend DeViSE in two directions: by leveraging visual context, and by reformulating it as a probabilistic model that allows coping with an imbalanced class distribution.

Visual context The intuitive principle that *some objects tend to be found in some contexts but not others*, is at the core of many works. In NLP, visual context of objects can be used to build efficient word representations [48]. In Computer Vision, it can be used to refine detection [8, 9] or segmentation [49] tasks.

Visual context can either be *low-level* (i.e. raw image pixels) or *high-level* (i.e. labeled objects). When visual context is exploited in the form of *low-level* information [41, 45, 42], it often consists of global image features. For instance, in [20], a Conditional Random Field is trained at combining low-level image features to assign to each pixel a class. In *high-level* approaches, the referential meaning of the context objects (i.e. class labels) is used. For example, [36] show that high-level context can be used at the post-processing level to reduce the ambiguities of a pre-learned object classification model,

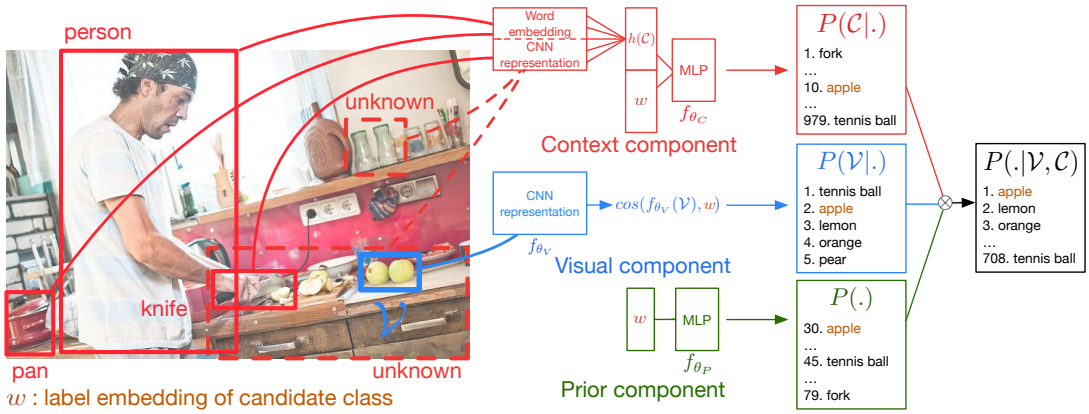


Figure 1: The goal is to find the class of the object contained within the blue region $\mathcal{V}$. Its context is formed of labeled objects from the source domain (red plain boxes) and of unlabeled object from the target domain (red dashed boxes).

by leveraging co-occurrence patterns between objects that are computed from the training set. Moreover, [47] study the role of context to classify objects: they investigate the importance of contextual indicators, such as object co-occurrence, relative scale and spatial relationships, and find that contextual information can sometimes be more informative than direct visual cues from objects. Spatial relations between objects can also be used in addition to co-occurrences, as in [18]. In [6], co-occurrences are computed using external information collected from web documents. The model classifies all objects jointly; it gives an inference method enabling a balance between an image coherence term (given by an image classifier) and a semantic term (given by a co-occurrence matrix). However, the approach is fully supervised, and this setting cannot be applied to ZSL.

In conclusion, while many works in NLP and Computer Vision show the importance of visual context, its use in ZSL remains a challenge, that we propose to tackle in this paper.

3 Context-aware ZSL

Let $\mathcal{O}$ be the set of all object classes, divided in classes from the *source domain* $\mathcal{S}$ and classes from the *target domain* $\mathcal{T}$. The goal of our approach — *context-aware ZSL* — is to determine the class $i \in \mathcal{T}$ of an object contained in an image I, given its visual appearance $\mathcal{V}$ and its visual context $\mathcal{C}$. The image I is annotated with bounding boxes, each containing an object. Given the zone $\mathcal{V}$, the context $\mathcal{C}$ consists of the surrounding objects in the image. Their classes can either belong to the source domain ($\mathcal{C} \cap \mathcal{S}$) or to the target domain ($\mathcal{C} \cap \mathcal{T}$). Note that the class of an object of $\mathcal{C} \cap \mathcal{T}$ is not accessible in ZSL, only its visual appearance is.

3.1 Model overview

We tackle this task by modeling the conditional probability $P(i|\mathcal{V},\mathcal{C})$ of a class i given both the visual appearance $\mathcal{V}$ and the visual context $\mathcal{C}$ of the object of interest. Given the absence of data in the target domain, we need to limit the complexity of the model, for generalizability's purpose. Accordingly, we suppose that $\mathcal{V}$ and $\mathcal{C}$ are conditionally independent given the class i — we show in the experiments (section 5) that this hypothesis is acceptable. This hypothesis leads to the following expression:

$$P(i|\mathcal{V},\mathcal{C}) \propto P(\mathcal{V}|i)P(\mathcal{C}|i)P(i) \quad (1)$$

where each conditional probability expresses the probability of either the visual appearance $\mathcal{V}$ or the context $\mathcal{C}$ given class i, and $P(i)$ denotes the prior distribution of the dataset.

The intuition behind our approach is illustrated in Figure 1, where the blue box contains the object of interest. Here, the class is *apple*, which belongs to the target domain $\mathcal{T}$. The visual component, which focuses on the zone $\mathcal{V}$, recognizes a *tennis ball* due to its yellow and round appearance; *apple* is ranked second. The prior component indicates that *apple* is slightly more frequent than *tennis ball*, but the frequency discrepancy may not be high enough to change the prediction of the visual component. In that case, the context component is discriminant: it ranks objects that are likely to be found in a kitchen, and reveals that an *apple* is far more likely to be found than a *tennis ball* in this context.

Precisely modeling $P(\mathcal{C}|.)$, $P(\mathcal{V}|.)$ and $P(.)$ is challenging due to the ZSL setting. Indeed, these distributions cannot be computed for classes of the target domain because of the absence of corresponding training data. Thus, to transfer the knowledge acquired

from the source domain to the target domain, we use a common semantic space, namely *Word2Vec* [31], where source and target class labels are embedded as vectors of $\mathbb{R}^d$, with d the dimension of the space. It is worth noting that we propose to separately learn the prior class distribution $P(.)$ with a ranking loss (in subsection 3.3). This allows dealing with imbalanced datasets, in contrast to ZSL models like DeViSE [13]. This intuition is experimentally validated in subsection 5.2.

3.2 Description of the components

Due to both the ZSL setting and the variety of possible context and/or visual appearance of objects, it is not possible to estimate directly the different probabilities of Equation 1. Hence, in what follows, we estimate quantities related to $P(\mathcal{C}|.)$, $P(\mathcal{V}|.)$ and $P(.)$ using parametric energy functions [26]. These quantities are learned separately, as described in subsection 3.3. Finally, we explain how we combine them to produce the global probability $P(.|\mathcal{C}, \mathcal{V})$ in subsection 3.4.

Visual component The visual component models $P(\mathcal{V}|i)$ by computing the compatibility between the visual appearance $\mathcal{V}$ of the object of interest, and the semantic representation w_i of the class i.

Following previous ZSL works based on cross-modal projections [13, 4], we introduce f_{θ_V}, a parametric function mapping an image to the semantic space: $f_{\theta_V}(\mathcal{V}) = W_V.\texttt{CNN}(\mathcal{V}) + b_V \in \mathbb{R}^d$ where $\texttt{CNN}(\mathcal{V})$ is a vector in $\mathbb{R}^{d_{\text{visual}}}$, output by a pretrained CNN truncated at the penultimate layer, W_V is a projection matrix ($\in \mathbb{R}^{d \times d_{\text{visual}}}$) and b_V a bias vector — in our experiments, $d_{\text{visual}} = 2048$. The probability that the image region $\mathcal{V}$ corresponds to the class i is set to be proportional to the cosine similarity between the projection $f_{\theta_V}(\mathcal{V})$ of $\mathcal{V}$ and the semantic representation w_i of the class i:

$$\log P(\mathcal{V}|i; \theta_V) \propto \cos(f_{\theta_V}(\mathcal{V}), w_i) := \log \widetilde{P}_{visual} \quad (2)$$

Context component The context component models $P(\mathcal{C}|i)$ by computing a compatibility score between the visual context $\mathcal{C}$, and the semantic representation w_i of the class i. More precisely, the conditional probability is written:

$$\log P(\mathcal{C}|i; \theta_C) \propto f_{\theta_C}(\mathcal{C}, w_i) = f_{\theta^1_C}\big(h_{\theta^2_C}(\mathcal{C}) \oplus w_i\big) := \log \widetilde{P}_{context} \quad (3)$$

where $h_{\theta^2_C}(\mathcal{C}) \in \mathbb{R}^d$ is a vector representing the context, $\theta_C = \{\theta^1_{\mathcal{C}}; \theta^2_{\mathcal{C}}\}$ are parameters to learn, and $\oplus$ is the concatenation operator. To take non-linear and high-order interactions between $h_{\theta^2_C}(\mathcal{C})$ and w_i into account, $f_{\theta^1_C}$ is modeled by a 2-layer Perceptron. We found that concatenating $h_{\theta^2_C}(\mathcal{C})$ with w_i leads to better results than a cosine similarity, as done in Equation 2 for the visual component.

To specify the modeling of $h_{\theta^2_C}(\mathcal{C})$, we propose various *context models* depending on which context objects are considered and how they are represented. Specifically, a context model is characterized by (a) the domain of context objects that are considered (i.e. source $\mathcal{S}$ or target $\mathcal{T}$) and (b) the way these objects are represented, either by a textual representation of their class label or by a visual representation of their image regions. Accordingly, we distinguish:

• The *low-level* (L) approach that computes a representation from the image region $\mathcal{V}_k$ of a context object. This produces the following context models:

$$S_L = \{W_C\texttt{CNN}(\mathcal{V}_k) + b_C | k \in \mathcal{C} \cap \mathcal{S}\}$$
$$T_L = \{W_C\texttt{CNN}(\mathcal{V}_k) + b_C | k \in \mathcal{C} \cap \mathcal{T}\}$$

• The *high-level* (H) approach which considers semantic representations w_k of the class labels k of the context objects (only available for entities of the source domain). This produces context models:

$$S_H = \{w_k | k \in \mathcal{C} \cap \mathcal{S}\} \text{ and } T_H = \{w_k | k \in \mathcal{C} \cap \mathcal{T}\}$$

Note that T_H is not defined in the zero-shot setting, since class labels of objects from the target domain are unknown; yet it is used to define Oracle models (subsection 4.3).

These four basic sets of vectors can further be combined in various ways to form new context models (for instance: $S_L \cup T_L, S_H \cup S_L, S_H \cup S_L \cup T_L$, etc.). At last, $h_{\theta^2_C}$ averages the representations of these vectors to build a global context representation. For example, $h_{\theta^2_C}(\mathcal{C}_{S_H \cup T_L})$ equals:

$$\frac{1}{|\mathcal{C}_{\mathcal{S}}| + |\mathcal{C}_{\mathcal{T}}|}\Big[\sum_{(i,\mathcal{V}_i) \in \mathcal{C}_{\mathcal{S}}} w_i + \sum_{(j,\mathcal{V}_j) \in \mathcal{C}_{\mathcal{T}}} \big(W_C.\texttt{CNN}(\mathcal{V}_j) + b_C\big)\Big]$$

where $|\cdot|$ denotes the cardinality of a set of vectors.

Prior component The goal of the prior component is to assess whether an entity is frequent or not in images. We estimate $P(i)$ from the semantic representation w_i of class i:

$$\log P(i; \theta_P) \propto f_{\theta_P}(w_i) := \log \widetilde{P}_{prior} \quad (4)$$

where f_{θ_P} is a 2-layer Perceptron that outputs a scalar.

3.3 Learning

In this section, we explain how we learn the energy functions f_{θ_C}, f_{θ_V} and f_{θ_P}. Each component (resp. context, visual, prior) of our model is assigned a training objective (resp. $\mathcal{L}_C$, $\mathcal{L}_V$, $\mathcal{L}_P$). As the components are independent by design, they are learned separately. This allows for a better generalization in the target domain, as shown experimentally (subsection 5.2). Besides, ensuring that some configurations are more likely than others motivates us to model each objective by a max-margin ranking loss, in which a positive configuration is assigned a lower energy than a negative one, following the *learning to rank* paradigm [44]. Unlike previous works [13], which are generally based on balanced datasets such as ImageNet and thus are not concerned with prior information, we want to avoid any bias coming from the imbalance of the dataset in $\mathcal{L}_C$ and $\mathcal{L}_V$, and learn the prior separately with $\mathcal{L}_P$. In other terms, the visual (resp. context) component should focus exclusively on the visual appearance (resp. visual context) of objects. This is done with a careful sampling strategy of the negative examples within the ranking objectives, that we detail in the following. To the best of our knowledge, such a discussion relative to prior modeling in learning objectives — which is, in our view, paramount in imbalanced datasets such as Visual Genome — has not been done in previous research.

Positive examples are sampled among entities of the source domain from the data distribution $P^\star$: they consist in a single object for $\mathcal{L}_P$, an object/box pair for $\mathcal{L}_V$, an object/context pair for $\mathcal{L}_C$. To sample negative examples j from the source domain, we distinguish two ways:

(1) For the prior objective $\mathcal{L}_P$, negative object classes are sampled from the *uniform* distribution U:

$$\mathcal{L}_P = \mathop{\mathbb{E}}_{i\sim P^\star} \mathop{\mathbb{E}}_{j\sim U} \lfloor \gamma_P - f_{\theta_P}(w_i) + f_{\theta_P}(w_j) \rfloor_+ \quad (5)$$

Noting $\Delta_{ji} := f_{\theta_P}(w_j) - f_{\theta_P}(w_i)$, the contribution of two given objects i and j to this objective is:

$$P^\star(i) \lfloor \gamma_P + \Delta_{ji} \rfloor_+ + P^\star(j) \lfloor \gamma_P - \Delta_{ji} \rfloor_+$$

If $P^\star(i) > P^\star(j)$, i.e. when object class i is more frequent than object class j, this term is minimized when $\Delta_{ji} = -\gamma_P$, i.e. $f_{\theta_P}(w_i) = f_{\theta_P}(w_j) + \gamma_P > f_{\theta_P}(w_j)$. Thus, $\widetilde{P}_{prior}(.;\theta_P)$ captures prior information, as it learns to rank objects based on their frequency.

(2) For the visual and context objectives, negative object classes are sampled from the prior distribution $P^\star(.)$:

$$\mathcal{L}_V = \mathop{\mathbb{E}}_{i,\mathcal{V}\sim P^\star} \mathop{\mathbb{E}}_{j\sim P^\star} \lfloor \gamma_V - f_{\theta_V}(\mathcal{V})^\top w_i + f_{\theta_V}(\mathcal{V})^\top w_j \rfloor_+ \quad (6)$$

$$\mathcal{L}_C = \mathop{\mathbb{E}}_{i,\mathcal{C}\sim P^\star} \mathop{\mathbb{E}}_{j\sim P^\star} \lfloor \gamma_C - f_{\theta_C}(\mathcal{C}, w_i) + f_{\theta_C}(\mathcal{C}, w_j) \rfloor_+ \quad (7)$$

Similarly, the contribution of two given objects i, j and a context $\mathcal{C}$ to the objective $\mathcal{L}_C$ is:

$$P^\star(i)P^\star(j)\Big[P^\star(\mathcal{C}|i)\lfloor \gamma_V + f_{\theta_C}(\mathcal{C}, w_j) - f_{\theta_C}(\mathcal{C}, w_i)\rfloor_+ + P^\star(\mathcal{C}|j)\lfloor \gamma_V + f_{\theta_C}(\mathcal{C}, w_i) - f_{\theta_C}(\mathcal{C}, w_j)\rfloor_+\Big]$$

Minimizing this term does not depend on the relative order between $P^\star(i)$ and $P^\star(j)$; thus, $\widetilde{P}_{context}(\mathcal{C}|.;\theta_C)$ does not take prior information into account. Moreover, $P^\star(\mathcal{C}|i) > P^\star(\mathcal{C}|j)$ implies that $f_{\theta_C}(\mathcal{C}, w_i) > f_{\theta_C}(\mathcal{C}, w_j)$.

The alternative, as done in DeViSE [13], is to sample negative classes uniformly in the source domain in the objective $\mathcal{L}_V$. Thus, if the prior is uniform, DeViSE directly models $P(.|\mathcal{V})$; otherwise, $\mathcal{L}_V$ cannot be analyzed straightforwardly. Besides, the contributions of visual and prior information are mixed. However, we show that learning the prior separately and imposing the context (resp. visual) component to exclusively focus on contextual (resp. visual) information is more efficient (subsection 5.2).

3.4 Inference

In this section, we detail the inference process. The goal is to combine the predictions of the individual components of the model to form the global probability distribution $P(.|\mathcal{V},\mathcal{C})$. In subsection 3.3, we detailed how to learn the functions f_{θ_C}, f_{θ_V} and f_{θ_P}, from which $\log \widetilde{P}_{context}$, $\log \widetilde{P}_{visual}$ and $\log \widetilde{P}_{prior}$ are deduced respectively. However, the normalization constants in Equations 2, 3 and 4, which depend on the object class i in the general case, are unknown. As a simplifying hypothesis, we suppose that these normalization constants are scalars that we respectively note α_C, α_V and α_P. This leads to:

$$P(.|\mathcal{V},\mathcal{C}) = \underbrace{(\widetilde{P}_{context})^{\alpha_C}}_{P(\mathcal{C}|.)} \cdot \underbrace{(\widetilde{P}_{visual})^{\alpha_V}}_{P(\mathcal{V}|.)} \cdot \underbrace{(\widetilde{P}_{prior})^{\alpha_P}}_{P(.)} \quad (8)$$

To see whether this hypothesis is reasonable, we did some *post-hoc* analysis of one of our model, and plotted in Figure 2 the values $\log \widetilde{P}_{visual}$, $\log \widetilde{P}_{context}$ and $\log \widetilde{P}_{prior}$ for positive (red points) and negative (blue points) configurations $(i,\mathcal{V},\mathcal{C})$ of the test set of Visual Genome. We observe that positive and negative

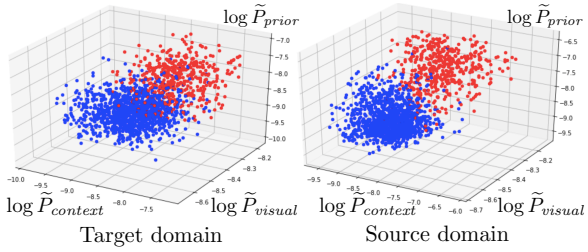


Figure 2: 3D visualization of the unnormalized log-probabilities of each component ($N = 500$). Context model $S_L \cup S_H \cup T_L$.

triplets are well separated, which empirically validates our initial hypothesis.

Hyper-parameters α_C, α_V and α_P are selected on the validation set to compute $P(.|\mathcal{C}, \mathcal{V})$. To build models that do not use a visual/contextual component, we simply select a subset of the probabilities and their respective hyperparameters. For example, $P(.|\mathcal{C}) = (\widetilde{P}_{context})^{\alpha_C}(\widetilde{P}_{prior})^{\alpha_P}$.

4 Experimental protocol

4.1 Data

To measure the role of context in ZSL, a dataset that presents annotated objects within a rich visual context is required. However, traditional ZSL datasets, such as AwA [11], CUB-200 [43] or LAD [50], are made of images that contain a unique object each, with no or very little surrounding visual context. We rather use Visual Genome [23], a large-scale image dataset (108K images) annotated at a fine-grained level (3.8M object instances), covering various concepts (105K unique object names). This dataset is of particular interest for our work, as objects have richly annotated contexts (31 object instances per image on average). In order to shape the data to our task, we randomly split the set of images of Visual Genome into train/validation/test sets (70%/10%/20% of the total size). To build the set $\mathcal{O}$ of all objects classes, we select classes which appear at least 10 times in Visual Genome and have an available *Word2vec* representation. $\mathcal{O}$ contains 4842 object classes; it amounts to 3.4M object instances in the dataset. This dataset is highly imbalanced as 10% of most represented classes amount to 84% of object instances. We define the *level of supervision* p_{sup} as the ratio of the size of the source domain over the total number of objects: $p_{\text{sup}} = |\mathcal{S}|/|\mathcal{O}|$. For a given p_{sup} ratio, the source $\mathcal{S}$ and target $\mathcal{T}$ domains are built by randomly splitting $\mathcal{O}$ accordingly. Every object is annotated with a bounding box and we use this supervision in our model for entities of both source and target domains.

4.2 Evaluation methodology

We adopt the conventional setting for ZSL, which implies entities to be retrieved only among the target domain $\mathcal{T}$. Besides, we also evaluate the performance of the model to retrieve entities of the source domain $\mathcal{S}$ (with models tuned on the target domain).

The model's prediction takes the form of a list of n classes, sorted by probability; the rank of the correct class in that list is noted r. Depending on the setting, n equals $|\mathcal{T}|$ or $|\mathcal{S}|$. We define the First Relevant (FR) metric with $\text{FR} = \frac{2}{n-1}(r-1)$. To further evaluate the performance over the whole test set, the Mean First Relevant (MFR) metric is used [17]. It is computed by taking the mean value of FR scores obtained on each image of the test set. Note that the factor $\frac{2}{n-1}$ rescales the metric such that the MFR score of a random baseline is 100%, while the MFR of a perfect model would be 0%. The MFR metric has the advantage to be interval-scale-based, unlike more traditional Recall@k metrics or Mean Reciprocal Ranks metrics [12], and thus can be averaged; this allows for meaningful comparison with a varying p_{sup}.

4.3 Scenarios and Baselines

Model scenarios Model scenarios depend on the information that is used in the probabilistic setting: $\varnothing$, $\mathcal{C}$, $\mathcal{V}$ or both $\mathcal{C}$ and $\mathcal{V}$. When contextual information is involved, a context model $\star$ is specified to represent $\mathcal{C}$, which we note $\mathcal{C}_\star$. The different context models are $\star \in \{S_H, S_L, T_L, S_L \cup T_L, S_H \cup T_L, S_L \cup S_H \cup T_L\}$. For clarity's sake, we note our model M. For example, $\text{M}(\mathcal{C}_{S_H \cup T_L}, \mathcal{V})$ models the probability $P(\mathcal{C}_{S_H \cup T_L}|.)P(\mathcal{V}|.)P(.)$ as explained in subsection 3.4, $\text{M}(\mathcal{V})$ models $P(\mathcal{V}|.)P(.)$, $\text{M}(\varnothing)$ models $P(.)$.

Oracles To evaluate upper-limit performances for our models, we define Oracle baselines where classes of target objects are used, which is not allowed in the zero-shot setting. Note that every Oracle leverages visual information.

• *True Prior:* This Oracle uses, for its prior component, the true prior distribution $P^\star(i) = \frac{\#i}{M}$ computed for all objects of both source and target domains on the full dataset, where $\#i$ is the number of instances of the i-th class in images and M is the total number of images.

• *Visual Bayes:* This Oracle uses $P^\star(.)$ for its prior component as well. Its context component uses co-occurrence statistics between objects computed on the full dataset: $P^{\text{im}}(\mathcal{C}|i) = \prod_{c\in\mathcal{C}} P_{\text{co-oc}}(c|i)$ where $P_{\text{co-oc}}(c|i) = \frac{\#(c,i)M}{\#c\#i}$ is the probability that objects c and i co-occur in images, with $\#(c,i)$ the number of co-occurrences of c and i.
• *Textual Bayes:* Inspired by [6], this Oracle is similar to Visual Bayes, except that its prior $P^{\text{text}}(.)$ and context component $P^{\text{text}}(.|\mathcal{C})$ are based on textual co-occurrences instead of image co-occurrences: $P_{\text{co-oc}}(c|i)$ is computed by counting co-occurrences of words c and i in windows of size 8 in the Wikipedia dataset, and $P^{\text{text}}(i)$ is computed by summing the number of instances of the i-th class divided by the total size of Wikipedia.
• *Semantic representations for all objects:* $\text{M}(\mathcal{C}_{S_H\cup T_H}, \mathcal{V})$ uses word embeddings of both source and target objects.

Baselines
• $\text{M}(\mathcal{C} \oplus \mathcal{V})$: To study the validity of the hypothesis about the conditional independence of $\mathcal{C}$ and $\mathcal{V}$, we introduce a baseline where we directly model $P(\mathcal{C}, \mathcal{V}|.)P(.)$. To do so, we replace, in the expression of $\mathcal{L}_V$ (Equation 6), $f_{\theta_V}(\mathcal{V})$ by the concatenation of $h(\mathcal{C})$ and $f_{\theta_V}(\mathcal{V})$ projected in $\mathbb{R}^d$ with a 2-layer Perceptron.
• DeViSE($\mathcal{V}$): To evaluate the impact of our Bayesian model (Equation 1) and our sampling strategy (subsection 3.3), we compare against DeViSE [13]. DeViSE($\mathcal{V}$) is different from $\text{M}(\mathcal{V})$ because negative examples in $\mathcal{L}_V$ are uniformly sampled, and the prior $P(.)$ is not learned.
• DeViSE($\mathcal{C} \oplus \mathcal{V}$): similarly to $\text{M}(\mathcal{C} \oplus \mathcal{V})$, we define a baseline that does not rely on the conditional independence of $\mathcal{C}$ and $\mathcal{V}$, using the same sampling strategy as DeViSE.
• $\text{M}(\mathcal{C}_I, \mathcal{V})$: To understand the importance of context supervision, i.e. annotations of context objects (boxes and classes), we design a baseline where no context annotations are used. The context is the whole image without the zone $\mathcal{V}$ of the object, which is masked out. The associated context model is $\star = I$ with $h(\mathcal{C}_\mathcal{I}) = g_{\theta_I}(I \setminus \mathcal{V})$; g_{θ_I} is a parametric function to be learned. This baseline is inspired from [42], where global image features are used to refine the prediction of an image model.

4.4 Implementation details

For each objective $\mathcal{L}_C, \mathcal{L}_V$ and $\mathcal{L}_P$, at each iteration of the learning algorithm, 5 negative entities are sampled per positive example. Word representations are vectors of $\mathbb{R}^{300}$, learned with the Skip-Gram algorithm [31] on Wikipedia. Image regions are cropped, rescaled to (299×299), and fed to `CNN`, an Inception-v3 CNN [40], whose weights are kept fixed during training. This model is pretrained on ImageNet [10]. As a result, every ImageNet class that belongs to the total set of objects $\mathcal{O}$ was included in the source domain $\mathcal{S}$. Models are trained with Adam [21] and regularized with a L2-penalty; the weight of this penalty decreases when the level of supervision increases, as the model is less prone to overfitting. All hyper-parameters are cross-validated on classes of the target domain, on the validation set.

		Target domain $\mathcal{T}$			Source domain $\mathcal{S}$		
	p_{sup}	10%	50%	90%	10%	50%	90%
	Domain size	4358	2421	484	484	2421	4358
	Random	*100*	*100*	*100*	*100*	*100*	*100*
Models	$\text{M}(\varnothing)$	38.6	23.7	13.8	12.0	10.6	11.2
	$\text{M}(\mathcal{V})$	20.5	10.7	6.0	1.5	2.6	3.6
	$\text{M}(\mathcal{C}_{S_H})$	28.7	14.4	9.1	4.2	4.3	4.4
	$\text{M}(\mathcal{C}_{S_H}, \mathcal{V})$	18.1	9.0	5.2	1.1	1.9	2.4
	$\delta_\mathcal{C}$ (%)	*11.6*	*16.4*	*12.1*	*23.7*	*27.3*	*31.5*

Table 1: Evaluation of various information sources, with varying levels of supervision. MFR scores in %. δ_C is the relative improvement (in %) of $\text{M}(\mathcal{C}_{S_H}, \mathcal{V})$ over $\text{M}(\mathcal{V})$.

5 Results

5.1 The importance of context

In this section, we evaluate the contribution of contextual information, with varying levels of supervision p_{sup}. We fix a simple context model ($\star = S_H$) and report MFR results with $p_{\text{sup}} = 10, 50, 90\%$ in section 5 for every combination of information sources: $\varnothing$, $\mathcal{V}$, $\mathcal{C}$ and $(\mathcal{C}, \mathcal{V})$ — we observe similar trends for the other context models. Results highlight that contextual knowledge acquired from the source domain can be transferred to the target domain, as $\text{M}(\mathcal{C}_{S_H})$ significantly outperforms the *Random* baseline. As expected, it is not as useful as visual information: $\text{M}(\mathcal{V})$ $\text{M}(\mathcal{C}_{S_H})$, where means lower MFR scores, i.e. better performances. However, section 5 demonstrates that contextual and visual information are complementary: using $\text{M}(\mathcal{C}_{S_H}, \mathcal{V})$ outperforms both $\text{M}(\mathcal{C}_{S_H})$ and $\text{M}(\mathcal{V})$. Interestingly, as the learned prior model $\text{M}(\varnothing)$ is also able to generalize, we show that visual frequency can somehow be learned from textual semantics, which extends previous work where word embeddings were shown to be a good predictor of textual frequency [39].

When p_{sup} increases, we observe that all models are better at retrieving objects of the target domain (i.e.

	Model	Probability	$\mathcal{T}$	$\mathcal{S}$
Oracles	*Textual Bayes*	$P^{\text{text}}(\mathcal{C}\|.)P(\mathcal{V}\|.)P^{\text{text}}(.)$	*14.54*	*6.73*
	$\text{M}(\mathcal{C}_{S_H \cup T_H}, \mathcal{V})$	$P(\mathcal{C}_{S_H \cup T_H}\|.)P(\mathcal{V}\|.)P(.)$	*7.57*	*2.53*
	True Prior	$P(\mathcal{V}\|.)P^{\star}(.)$	*4.92*	*2.63*
	Visual Bayes	$P^{\text{im}}(\mathcal{C}\|.)P(\mathcal{V}\|.)P^{\star}(.)$	*3.40*	*2.11*
Baselines	$\text{DeViSE}(\mathcal{V})$	$P(.\|\mathcal{V})$	10.73	3.62
	$\text{DeViSE}(\mathcal{C}_{S_H} \oplus \mathcal{V})$	$P(.\|\mathcal{C}_{S_H}, \mathcal{V})$	10.11	3.11
	$\text{M}(\mathcal{C}_{S_H} \oplus \mathcal{V})$	$P(\mathcal{C}_{S_H}, \mathcal{V}\|.)P(.)$	10.07	1.85
	$\text{M}(\mathcal{C}_I, \mathcal{V})$	$P(\mathcal{C}_I\|.)P(\mathcal{V}\|.)P(.)$	9.19	2.13
Our models	$\text{M}(\mathcal{V})$	$P(\mathcal{V}\|.)P(.)$	10.72	2.64
	$\text{M}(\mathcal{C}_{S_L}, \mathcal{V})$	$P(\mathcal{C}_{S_L}\|.)P(\mathcal{V}\|.)P(.)$	9.01	2.05
	$\text{M}(\mathcal{C}_{T_L}, \mathcal{V})$	$P(\mathcal{C}_{T_L}\|.)P(\mathcal{V}\|.)P(.)$	9.00	2.13
	$\text{M}(\mathcal{C}_{S_H}, \mathcal{V})$	$P(\mathcal{C}_{S_H}\|.)P(\mathcal{V}\|.)P(.)$	8.96	1.92
	$\text{M}(\mathcal{C}_{S_L \cup T_L}, \mathcal{V})$	$P(\mathcal{C}_{S_L \cup T_L}\|.)P(\mathcal{V}\|.)P(.)$	8.60	1.93
	$\text{M}(\mathcal{C}_{S_H \cup T_L}, \mathcal{V})$	$P(\mathcal{C}_{S_H \cup T_L}\|.)P(\mathcal{V}\|.)P(.)$	8.52	1.86
	$\text{M}(\mathcal{C}_{S_H \cup S_L \cup T_L}, \mathcal{V})$	$P(\mathcal{C}_{S_H \cup S_L \cup T_L}\|.)P(\mathcal{V}\|.)P(.)$	**8.31**	**1.79**

Table 2: MFR performances (given in %) for all baselines and scenarios. $p_{\text{sup}} = 50\%$. Oracle results, written in italics, are not taken into account to determine the best scores.

MFR decreases), which is intuitive because models are trained on more data and thus generalize better to recognize entities from the target domain. Besides, when p_{sup} increases, the context is also more abundant. This explains: (1) the decreasing MFR values for model $\text{M}(\mathcal{C}_{S_H})$ on $\mathcal{T}$, (2) the increasing relative improvement δ_C of $\text{M}(\mathcal{C}_{S_H}, \mathcal{V})$ over $\text{M}(\mathcal{V})$ on $\mathcal{S}$. However, on the target domain, we note that δ_C does not monotonously increase with p_{sup}. A possible explanation is that the visual component improves faster than the context component, so the relative contribution brought by context to the final model $\text{M}(\mathcal{C}_{S_H}, \mathcal{V})$ decreases after $p_{\text{sup}} = 50\%$. Since the highest relative improvement δ_C (in $\mathcal{T}$) is attained with $p_{\text{sup}} = 50\%$, we fix the level of supervision to this value in the rest of the experiments; this amounts to 2421 classes in both source and target domains.

5.2 Modeling contextual information

In this section, we compare the different context models; results are reported in subsection 5.2. First, underlying hypotheses of our model are experimentally tested. (1) Modeling context and prior information with semantic representations (models $\text{M}(\mathcal{C}_\star, \mathcal{V})$) is far more efficient than using direct textual co-occurrences, as shown by the *Textual Bayes* baseline, which is the weaker model despite being an Oracle. (2) Moreover, we show that the hypothesis on the conditional independence of $\mathcal{C}$ and $\mathcal{V}$ is acceptable, as separately modeling $\mathcal{C}$ and $\mathcal{V}$ gives better results than jointly modeling them (i.e. $\text{M}(\mathcal{C}_{S_H}, \mathcal{V})$ $\text{M}(\mathcal{C}_{S_H} \oplus \mathcal{V})$). (3) Furthermore, we observe that our approach $\text{M}(\mathcal{V})$ is more efficient to capture the imbalanced class distribution of the source domain, compared to $\text{DeViSE}(\mathcal{V})$; indeed, *True Prior* $\approx \text{M}(\mathcal{V})$, whereas *True Prior* $\text{DeViSE}(\mathcal{V})$ on $\mathcal{S}$. Even if the improvement is only significant for the source domain $\mathcal{S}$, it indicates that separately using information sources is clearly a superior approach to further integrate contextual information.

Second, as observed in the case of the context model S_H (subsection 5.1), using contextual information is always beneficial. Indeed, all models with context $\text{M}(\mathcal{C}_\star, \mathcal{V})$ improve over $\text{M}(\mathcal{V})$ — which is the model with no contextual information — both on target and source domains. In more details, we observe that performances increase when additional information is used: (1) when the bounding boxes annotations are available: all of our models that use both $\mathcal{C}$ and $\mathcal{V}$ outperform the baseline $\text{M}(\mathcal{C}_I, \mathcal{V})$, which could also be explained by the useless noise outside the object boxes in the image and the difficulty of computing a global context from raw image, (2) when context objects are labeled and high-level features are used instead of low-level features, e.g. $S_H S_L$ and $S_H \cup T_H S_H \cup T_L$, (3) when more context objects are considered (e.g. $S_L \cup T_L S_L$), (4) when low-level information is used complementarily to high-level information (e.g. $S_L \cup S_H \cup T_L S_L \cup T_L$). As a result, the best performance is attained for $\text{M}(\mathcal{C}_{S_L \cup S_H \cup T_L}, \mathcal{V})$, with a 22% (resp. 32%) relative improvement in the target (resp. source) domain compared to $\text{M}(\mathcal{V})$.

We note that there is still room for improvement to approach ground-truth distributions for objects of the target domain (e.g. towards word embeddings that better capture visual context). Indeed, even if our models outperform *True Prior* and *Visual Bayes* on the source domain, these Oracle baselines are still better on the target domain, hence showing that learning the visual context of objects from textual data is challenging.

5.3 Qualitative Experiments

To gain a deeper understanding of contextual information, we compare in Figure 3 the predictions of $\text{M}(\mathcal{V})$ and the global model $\text{M}(\mathcal{C}, \mathcal{V})$. We randomly select five classes of the target domain and plot, for all instances of these classes in the test set of Visual Genome, the distribution of the predicted ranks of the correct class (in percentage); we also list the classes that appear the most in the context of these classes. We observe that, for certain classes (*player*, *handle* and *field*), contextual information helps to refine the predictions; for others (*house* and *dirt*), contextual information degrades the quality of the predictions.

First, we can outline that visual context can guide the model towards a more precise prediction. For exam-

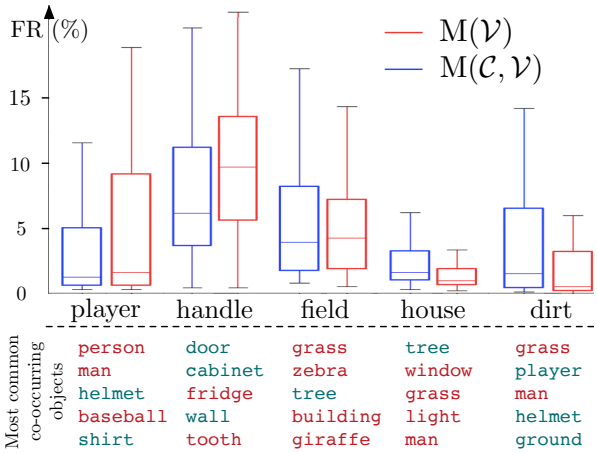


Figure 3: Boxplot representing the distribution of the correct ranks (First Relevant in %) for five randomly selected classes of the target domain, with the context model $S_L \cup S_H \cup T_L$. Below are listed, by order of frequency, the classes that co-occur the most with the object of interest (classes of $\mathcal{T}$ in green; $\mathcal{S}$ in red).

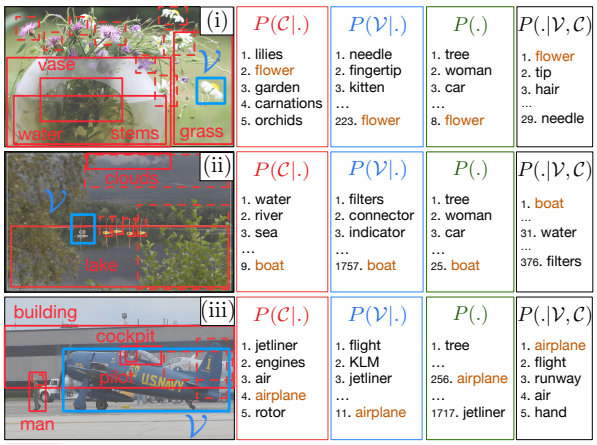


Figure 4: Qualitative examples where the global model $\mathrm{M}(\mathcal{C}_{S_L \cup S_H \cup T_L}, \mathcal{V})$ ranks the correct class first (in $\mathcal{T}$ only).

ple, a *player*, without context, could be categorized as *person*, *man* or *woman*; but visual context provides important complementary information (e.g, *helmet*, *baseball*, *bat*...) that grounds *person* in a sport setting, and thus suggests that the *person* could be playing. Visual context is also particularly relevant when the object of interest has a generic shape. For example, *handle*, without context, is visually similar to many round objects; but it is the presence of objects like *door* or *fridge* in the context that helps determine the nature of the object of interest.

To get a better insight on the role of context, we cherry-picked examples where the visual or the prior component is inacurrate and the context component is able to counterbalance the final prediction (Figure 4). In (i), for example, the visual component ranks *flower* at position 223. However, the context component assesses *flower* to be highly probable in this context, due to the presence of source objects like *vase*, *water*, *stems* or *grass*, but also target objects like the other flowers around. At the inference phase, probabilities are aggregated and *flower* is ranked first.

It is worth noting that our work is not without limitations. Indeed, some classes (such as *house* and *dirt*) have a wide range of possible contexts; in these cases, context is not a discriminating factor. This is confirmed by a complementary analysis: the Spearman correlation between the number of unique context objects and δ_C, the relative gain of $\mathrm{M}(\mathcal{C}_{S_H}, \mathcal{V})$ over $\mathrm{M}(\mathcal{V})$, is $\rho = -0.31$. In other terms, contextual information is useful for specific objects, which appear in particular contexts; for objects that are too generic, adding contextual information can be a source of noise.

6 Conclusion

In this paper, we introduced a new approach for ZSL: *context-aware ZSL*, along with a corresponding model, using complementary contextual information that significantly improves predictions. Possible extensions could include spatial features of objects, and, more importantly, removing the dependence on the detection of object boxes to make it fully applicable to real-world images. Finally, designing grounded word embeddings that include more visual context information would also benefit such models.

7 Acknowledgment

This work is partially funded by the Labex SMART (ANR-11-LABX-65) and MUSTER (PCIN-2015-226).

References

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for image classification. *TPAMI*, 2016.

[2] Z. Akata, S. E. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *CVPR*, 2015.

[3] L. J. Ba, K. Swersky, S. Fidler, and R. Salakhutdinov. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *ICCV*, 2015.

[4] A. Bansal, K. Sikka, G. Sharma, R. Chellappa, and A. Divakaran. Zero-shot object detection. In *ECCV*, 2018.

[5] S. Bell, C. L. Zitnick, K. Bala, and R. B. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016.

[6] S. Bengio, J. Dean, D. Erhan, E. Ie, Q. V. Le, A. Rabinovich, J. Shlens, and Y. Singer. Using web co-occurrence statistics for improving image categorization. *CoRR*, abs/1312.5697, 2013.

[7] M. Bucher, S. Herbin, and F. Jurie. Improving semantic embedding consistency by metric learning for zero-shot classiffication. In *ECCV*, 2016.

[8] Q. Chen, Z. Song, J. Dong, Z. Huang, Y. Hua, and S. Yan. Contextualizing object detection and classification. *TPAMI*, 2015.

[9] W. Chu and D. Cai. Deep feature based contextual model for object detection. *Neurocomputing*, 2018.

[10] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[11] A. Farhadi, I. Endres, D. Hoiem, and D. A. Forsyth. Describing objects by their attributes. In *CVPR*, 2009.

[12] M. Ferrante, N. Ferro, and S. Pontarollo. Are IR evaluation measures on an interval scale? In *SIGIR*, 2017.

[13] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013.

[14] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong. Learning multimodal latent attributes. *TPAMI*, 2014.

[15] Y. Fu, Y. Yang, T. M. Hospedales, T. Xiang, and S. Gong. Transductive multi-label zero-shot learning. *CoRR*, abs/1503.07790, 2015.

[16] Z. Fu, T. A. Xiang, E. Kodirov, and S. Gong. Zero-shot object recognition by semantic manifold distance. In *CVPR*, 2015.

[17] N. Fuhr. Some common mistakes in IR evaluation, and how they can be avoided. *SIGIR Forum*, 2017.

[18] C. Galleguillos, A. Rabinovich, and S. J. Belongie. Object categorization using co-occurrence, location and appearance. In *CVPR*, 2008.

[19] Z. S. Harris. Distributional structure. *Word*, 1954.

[20] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *CVPR*, 2004.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[22] E. Kodirov, T. Xiang, and S. Gong. Semantic autoencoder for zero-shot learning. In *CVPR*, 2017.

[23] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2017.

[24] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *TPAMI*, 2014.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *IEEE*, 1998.

[26] Y. LeCun, S. Chopra, and R. Hadsell. A tutorial on energy-based learning. *Predicting Structured Data*, 2006.

[27] J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *CVPR*, 2011.

[28] Y. Long, L. Liu, L. Shao, F. Shen, G. Ding, and J. Han. From zero-shot learning to conventional supervised classification: Unseen visual data synthesis. In *CVPR*, 2017.

[29] T. Mensink, E. Gavves, and C. Snoek. Costa: Co-occurrence statistics for zero-shot classification. *CVPR*, 2014.

[30] T. Mensink, J. J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *ECCV*, 2012.

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[32] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. *CoRR*, abs/1312.5650, 2013.

[33] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[34] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

[35] R. Qiao, L. Liu, C. Shen, and A. van den Hengel. Less is more: Zero-shot learning from online textual documents with noise suppression. In *CVPR*, 2016.

[36] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. J. Belongie. Objects in context. In *ICCV*, 2007.

[37] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. *CoRR*, abs/1802.01548, 2018.

[38] B. Romera-Paredes and P. H. S. Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, 2015.

[39] A. M. J. Schakel and B. J. Wilson. Measuring word significance using distributed representations of words. *CoRR*, abs/1508.02297, 2015.

[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[41] A. Torralba. Contextual priming for object detection. *IJCV*, 2003.

[42] A. Torralba, K. P. Murphy, and W. T. Freeman. Using the forest to see the trees: exploiting context for visual object detection and localization. *ACM*, 2010.

[43] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[44] J. Weston, S. Bengio, and N. Usunier. WSABIE: scaling up to large vocabulary image annotation. In *IJCAI*, 2011.

[45] L. Wolf and S. M. Bileschi. A critical view of context. *IJCV*, 2006.

[46] Y. Xian, Z. Akata, G. Sharma, Q. N. Nguyen, M. Hein, and B. Schiele. Latent embeddings for zero-shot classification. In *CVPR*, 2016.

[47] R. Yu, X. Chen, V. I. Morariu, and L. S. Davis. The role of context selection in object detection. In *BMVC*, 2016.

[48] É. Zablocki, B. Piwowarski, L. Soulier, and P. Gallinari. Learning Multi-Modal Word Representation Grounded in Visual Context. In *AAAI*, 2018.

[49] H. Zhang, K. J. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal. Context encoding for semantic segmentation. In *CVPR*, 2018.

[50] B. Zhao, B. Chang, Z. Jie, and L. Sigal. Modular generative adversarial networks. *CoRR*, abs/1804.03343, 2018.

[51] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.

Segmentation adaptative d'image avec un nombre de classes efficace en utilisant l'algorithme Expectation-Maximisation pour modèle de mélange de processus de Dirichlet tronqué

Hong-Phuong Dang et Myriam Vimond

CREST - CNRS UMR 9194, ENSAI Bruz

28 mai 2019

Résumé

La segmentation d'image consiste à partitionner les pixels d'une image en différents groupes homogènes. Cela peut être vu comme un problème de regroupement (*clustering*). L'une des questions de *clustering* est de définir le nombre de classes pertinent qui peut s'adapter à la complexité des données. Les approches bayésiennes non paramétriques (BNP) peuvent éviter ce choix non trivial. Le modèle de mélanges par processus de Dirichlet (DPM) est une approche BNP populaire pour les problèmes de *clustering* avec un nombre de classes inconnu puisqu'il existe des inférences relativement simples pour ces modèles. Dans un DPM, le paramètre de concentration joue un rôle crucial dans la sélection du nombre de classes. Ce papier propose une procédure permettant de calculer l'estimateur maximum *a posteriori* (MAP) à partir de la représentation *stick-breaking* tronquée d'un modèle DPM et d'un algorithme Expectation-Maximisation (EM). Cette procédure inclut l'apprentissage du paramètre de concentration. Les résultats sur les données simulées et la segmentation d'image illustrent la pertinence de la méthode proposée.

Mots-clef : Classification non supervisée, segmentation d'image, Bayésien non paramétrique, processus de Dirichlet, modèle de mélanges, algorithme EM.

1 Introduction

La classification non supervisée ou le regroupement (*clustering*) est une technique d'apprentissage automatique qui consiste à partitionner un ensemble de données en regroupant des données ayant des caractéristiques similaires dans la même classe. De nombreuses méthodes de *clustering* ont été développées et sont classées dans les familles suivantes [1, 2] : méthodes de partitionnement, hiérarchiques, basées sur la densité, de grille et à modèles. Les techniques de *clustering* ont été utilisées dans de nombreuses disciplines telles que les études de marché, la reconnaissance de formes, l'analyse de données et le traitement d'images. En particulier, la segmentation d'image est l'un des problèmes fondamentaux du traitement d'images qui utilise souvent des méthodes de *clustering*. Elle vise à partitionner une image en différentes régions où les pixels se comportent de la même manière. Ses applications pratiques sont sur l'imagerie médicale [3], la vidéo surveillance [4] *etc.*

Cependant, déterminer le nombre de classes K reste un problème difficile. K est souvent considéré comme connu [5–7] et spécifié sur une connaissance a priori ou des heuristiques. K peut aussi être choisi en fonction d'un critère prédéfini [1, 8]. Les critères traditionnels mesurent la qualité du *clustering* (l'indice de Hartigan, l'indice Silhouette, la statistique Gap, analyse de stabilité, etc.) ou pénalisent la vraisemblance [9] (AIC, BIC, ICL, etc.). Par contre, il est nécessaire de tester plusieurs modèles pour différentes valeurs de K afin de choisir la valeur qui répond le mieux aux critères demandés. Parmi les différentes approches qui ont abordé cette question, la plus populaire dans le cadre bayésien est la famille des méthodes de Monte-Carlo par chaînes de Markov (MCMC) à sauts réversibles. Il consiste à utiliser des méthodes MCMC afin de simuler une chaîne de Markov de loi stationnaire qui est définie sur un espace particulier. Cet espace contient plusieurs sous-espaces de différents dimensions. La chaîne de Markov simulée doit pouvoir "sauter" d'un sous-espace dans un autre. Néanmoins, la charge de calcul de ces techniques reste élevé.

Pour aborder la question du nombre adaptatif de classes, les méthodes bayésiennes non paramétriques (BNP) sont très prometteuses. *Non paramétriques* ne veut pas dire que l'on ignore l'existence des paramètres des distributions. Les méthodes non paramétriques supposent que le modèle peut être caractérisé par un vecteur de paramètre de dimension potentiellement infinie. On va estimer ces paramètres ainsi que leur dimension en utilisant des lois *a priori* définies sur des espaces de distributions. Dans le cas de *clustering*, le nombre de classes dans un modèle BNP est potentiellement infini, mais le nombre (fini) K de classes actives est contrôlé par une loi *a priori* . Une classe est dite active si elle contient au moins une donnée. Par conséquent, un ensemble de N données a un maximum de $K = N$ de classes actives. Au lieu de comparer des modèles avec différentes valeurs de K, les modèles BNP s'adaptent à un seul modèle et choisissent K en fonction de la complexité des données.

Cet article étudie l'utilisation du modèle BNP de mélange qui est classé dans la famille des méthodes *clustering* basée sur des modèles probabilistes. En particulier, on s'intéresse au modèle de mélange par processus de Dirichlet (DPM) [10]. Le DP est une distribution qui dépend de deux paramètres et ses réalisations sont composées d'un nombre infini de composantes (classes). Les modèles DPM sont devenus populaires dans le traitement statistique du signal en raison de l'existence des algorithmes MCMC relativement simples pour l'inférence. Les travaux de [11, 12] ont démontré le potentiel de DPM pour la segmentation d'image sans nécessiter la connaissance préalable du nombre de classes. Cependant, dans ces approches, le paramètre de concentration α - l'un des deux paramètres de DPM - est fixé. Ce paramètre contrôle le nombre de classes actives. Dans l'esprit d'un modèle bayésien DPM, nous adoptons une version tronquée du DP avec les loi *a priori* sur tous les paramètres, y compris α [13]. L'objectif ici est de calculer l'estimateur MAP. Pour cela, l'algorithme Expectation-Maximisation [14] est dérivé pour l'inférence en raison de son coût de calcul inférieur à celui de, par exemple, l'approche MCMC.

La section 2 introduit le processus de Dirichlet. La section 3 présente l'intérêt d'utiliser le processus de Dirichlet pour les modèles de mélange. La section 4 formule le modèle proposé avant de décrire l'algorithme TDPM-EM pour l'inférence. La section 5.2 résume les résultats expérimentaux sur les données simulées et la segmentation d'image afin d'illustrer la pertinence de la méthode proposée. La section 6 conclut et évoque certaines orientations pour les futurs travaux.

2 Processus de Dirichlet (DP)

Le DP [15] se définit comme une distribution sur l'espace des distributions. Soit $\mathbb{G}_0$ une mesure de probabilité sur un espace mesurable $(\Theta, \mathcal{A})$. Une distribution de probabilité $\mathbb{G}$ est dite distribuée selon un DP de distribution de base G_0 et de paramètre de concentration α, noté $\mathbb{G} \sim \mathrm{DP}(\alpha, G_0)$, si pour toute partition mesurable $(A_1, \ldots, A_k)$ de Θ formant une partition de $\mathcal{A}$,

$$[\mathbb{G}(A_1), \ldots, \mathbb{G}(A_k)] \sim \mathcal{D}ir\left(\alpha G_0(A_1), \ldots, \alpha G_0(A_k)\right). \quad (1)$$

La réalisation $\mathbb{G}$ d'un DP est presque surement une mesure discrète et peut être représentée par [16]

$$\mathbb{G} = \sum_{k=1}^{\infty} \pi_k \delta_{\phi_k}. \quad (2)$$

Cette représentation *stick-breaking* contient deux suites aléatoires indépendantes (ϕ_k) et (π_k) définies par

$$\begin{aligned} \phi_k &\overset{i.i.d}{\sim} G_0 \\ \pi_k &= \nu_k \prod_{l=1}^{k-1} (1-\nu_l) \quad \text{où} \qquad \nu_k \overset{i.i.d}{\sim} \mathrm{Beta}(1, \alpha). \end{aligned} \quad (3)$$

Pour chaque réalisation du DP, la position des (ϕ_k) est déterminée par la distribution de base G_0, et les poids (π_k) sont définis selon un processus en fonction du paramètre de concentration α. Par la suite, nous utilisons une représentation *stick-breaking* pour la construction de la procédure d'inférence dans la section 4.

3 Mélanges par DP (DPM)

Les modèles DPM sont un type de modèles très populaire dans la littérature [17, 18]. Pour chaque $\theta \in \Theta$ considérons $x \to f(x|\theta)$ une densité de probabilité définie sur $\mathbb{R}^d$. Soit $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N) \in \mathbb{R}^{d \times N}$ la matrice des N observations supposées indépendantes et identiquement distribuées $(i.i.d)$ selon une loi de densité F inconnue qui est un mélange de densités $f(x|\theta)$

$$\mathbf{y}_i \overset{i.i.d}{\sim} F, \quad \forall i = 1, \ldots, N. \quad (4)$$

Dans ce cas, le DP permet de ne pas se limiter aux familles de distributions paramétriques où le nombre de composantes est fixé et fini (modèle de mélange fini). Cependant, les réalisations d'un DP sont discrètes (eq. (2)), ce qui ne correspond pas à une densité de probabilité continue. Une solution consiste à lisser la mesure discrète en utilisant un noyau $f(\mathbf{y} \mid \theta)$ [19]

$$F(\mathbf{y}) = \int_{\Theta} f(\mathbf{y}|\theta) \mathrm{d}\mathbb{G}(\theta), \quad (5)$$

Ici, le DP $\mathbb{G}$ s'interprète comme la loi d'un modèle de mélange possédant une infinité dénombrable de composantes. Les variables $\theta \in \Theta$ représentent les paramètres latents propres au mélange. Le modèle peut être réécrit selon un modèle Bayésien hiérarchique [10],

$$\begin{aligned} \mathbb{G} &\sim \mathrm{DP}(\alpha, G_0), \\ \theta_i \mid \mathbb{G} &\overset{i.i.d.}{\sim} \mathbb{G}, \\ \mathbf{y}_i \mid \theta_i &\overset{ind}{\sim} f(\cdot \mid \theta_i), \quad i = 1 \dots N. \end{aligned} \tag{6}$$

Comme évoqué ci-dessus, pour N observations, on a un maximum de $K = N$ de composantes actives. Ici, le paramètre α de DP contrôle le nombre de composantes actives du modèle : de grandes valeurs de α préjugent d'un nombre élevé de composantes actives. Le choix de α a donc un impact important sur les performances du modèle. La section 4 présente l'approche proposée qui estime simultanément α et les paramètres latents.

4 Méthode proposée

Pour simplifier, on se fixe le cadre du mélange de gaussiennes pour le modèle (6) : la densité $f(\mathbf{y}|\theta)$ est $\mathcal{N}(\mu, \Sigma)$ pour $\theta = (\mu, \Sigma)$. La distribution de base G_0 est une loi *a priori multinormal-Wishart* pour $\phi_k = (\mu_k, \Sigma_k)$, (*cf.* eq. (3) pour la définition de ϕ_k),

$$\Sigma_k \sim \mathrm{InvWishart}(s, S), \quad \mu_k|\Sigma_k \sim \mathcal{N}(m, \Sigma_k/\tau). \tag{7}$$

Pour inférer un modèle DPM, il existe principalement deux approches. La première exploite l'échantillonnage conditionnel selon le schéma d'urne de Polya en simulant à chaque itération le nombre de composantes actives. Ici, nous adoptons la seconde approche qui consiste à approcher le DP par une représentation *stick-breaking* tronquée. Soit $K_{\max} \geq 2$ un nombre maximum de composantes actives, par exemple $K_{\max} = N$. La représentation tronquée de $\mathbb{G}$ est

$$\mathbb{G} \simeq \sum_{k=1}^{K_{\max}} \pi_k \delta_{\phi_k}. \tag{8}$$

En ajoutant une loi *a priori* sur le paramètre α, le modèle DPM basé sur la représentation *stick-breaking* tronquée peut se réécrire sous la forme,

$$\alpha \sim \mathrm{Gamma}(a, b), \tag{9}$$

$$(\nu_k)_{k=1\dots K_{\max}-1} \overset{i.i.d}{\sim} \mathrm{Beta}(1, \alpha), \quad \nu_{K_{\max}} = 1 \tag{10}$$

$$(\pi_k)_{k=2\dots K_{\max}} = \nu_k \prod_{l=1}^{k-1} (1 - \nu_l), \quad \pi_1 = \nu_1 \tag{11}$$

$$z_1, \dots, z_N \mid \boldsymbol{\nu} \overset{i.i.d}{\sim} \boldsymbol{\pi}, \tag{12}$$

$$\boldsymbol{\phi} = (\phi_k)_{k=1\dots K_{\max}} \overset{i.i.d}{\sim} G_0, \tag{13}$$

$$\mathbf{y}_i \mid z, \phi \overset{ind}{\sim} f(\cdot \mid \phi_{z_i}), \; i = 1, \dots N \tag{14}$$

où $\boldsymbol{\pi} = (\pi_1, \dots, \pi_{K_{\max}})$ est le vecteur des poids défini à partir de $\boldsymbol{\nu} = (\nu_1, \dots, \nu_{K_{\max}})$. Ainsi les paramètres (a, b, s, S, m, τ) sont les hyperparamètres du modèle. Soulignons que ce modèle est une approximation paramétrique du modèle DPM.

L'estimateur MAP pour le modèle DPM consiste ici à maximiser la densité *a posteriori* de $(\boldsymbol{\nu}, \boldsymbol{\phi}, \alpha)$. Nous proposons un algorithme EM *double* utilisant la variable latente $\mathbf{z} = (z_1, \dots, z_N)$ décrit selon le schéma suivant à l'itération t :

$$(\boldsymbol{\nu}^{(t)}, \boldsymbol{\phi}^{(t)}) = \arg\max_{\boldsymbol{\nu}, \boldsymbol{\phi}} Q_1\left(\boldsymbol{\nu}, \boldsymbol{\phi} \,\middle|\, \boldsymbol{\nu}^{(t-1)}, \boldsymbol{\phi}^{(t-1)}, \alpha^{(t-1)}\right), \tag{15}$$

$$\alpha^{(t)} = \arg\max_{\alpha > 0} Q_2\left(\alpha \,\middle|\, \boldsymbol{\nu}^{(t)}, \boldsymbol{\phi}^{(t)}, \alpha^{(t-1)}\right), \tag{16}$$

où

$$\begin{aligned} &Q_1\left(\boldsymbol{\nu}, \boldsymbol{\phi} \,\middle|\, \boldsymbol{\nu}^{(t-1)}, \boldsymbol{\phi}^{(t-1)}, \alpha^{(t-1)}\right) \\ &\quad = \mathbb{E}\Big(\log \mathrm{p}(\mathbf{z}, \boldsymbol{\nu}, \boldsymbol{\phi}, \alpha^{(t-1)}|\mathbf{Y}) \,\Big|\, \mathbf{Y}, \boldsymbol{\nu}^{(t-1)}, \boldsymbol{\phi}^{(t-1)}, \alpha^{(t-1)}\Big), \\ &Q_2\left(\alpha \,\middle|\, \boldsymbol{\nu}^{(t)}, \boldsymbol{\phi}^{(t)}, \alpha^{(t-1)}\right) \\ &\quad = \mathbb{E}\left(\log \mathrm{p}(\mathbf{z}, \boldsymbol{\nu}^{(t)}, \boldsymbol{\phi}^{(t)}, \alpha|\mathbf{Y}) \,\middle|\, \mathbf{Y}, \boldsymbol{\nu}^{(t)}, \boldsymbol{\phi}^{(t)}, \alpha^{(t-1)}\right). \end{aligned}$$

Il est facile de montrer que cet algorithme fait croître de manière monotone la densité *a posteriori*, *i.e.*

$$\mathrm{p}\Big(\mathbf{z}, \boldsymbol{\nu}^{(t)}, \boldsymbol{\phi}^{(t)}, \alpha^{(t)} \,|\mathbf{Y}\Big) \geq \mathrm{p}\Big(\mathbf{z}, \boldsymbol{\nu}^{(t-1)}, \boldsymbol{\phi}^{(t-1)}, \alpha^{j-1} \,|\mathbf{Y}\Big). \tag{17}$$

Dans notre modèle avec des composantes gaussiennes, les fonctions Q_1 et Q_2 atteignent leur maxima respectifs aux points suivants,

$$\mu_k^{(t)} = \frac{m\tau + \sum_{i=1}^N y_i \omega_{i,k}^{(t-1)}}{\tau + \sum_{i=1}^N \omega_{i,k}^{(t-1)}}, \tag{18}$$

$$\begin{aligned} \Sigma_k^{(t)} = \Bigg(&\sum_{i=1}^N (y_i - \mu_k^{(t)})(y_i - \mu_k^{(t)})^{\mathrm{T}} \omega_{i,k}^{(t-1)} + S \\ &+ \tau(\mu_k^{(t)} - m)(\mu_k^{(t)} - m)^{\mathrm{T}}\Bigg) / (s + d + 3 + \sum_{i=1}^N \omega_{i,k}^{(t-1)}), \end{aligned} \tag{19}$$

$$\nu_k^{(t)} = \frac{\sum_{i=1}^N \omega_{i,k}^{(t-1)}}{\alpha^{(t-1)} - 1 + \sum_{\ell=k}^{K_{\max}} \sum_{i=1}^N \omega_{i,k}^{(t-1)}}, \tag{20}$$

$$\alpha^{(t)} = \frac{K_{\max} + a - 2}{b - \sum_{k=1}^{K_{\max}-1} \log\left(1 - \nu_k^{(t)}\right)}, \tag{21}$$

où

$$\begin{aligned} \omega_{i,k}^{(t-1)} &= \mathbb{P}\left(z_i = k | y, \boldsymbol{\nu}^{(t-1)}, \boldsymbol{\phi}^{(t-1)}, \alpha^{(t-1)}\right) \\ &= \frac{\pi_k^{(t-1)} f(x_i|\theta_k^{(t-1)})}{\sum_{\ell=1}^{K_{\max}} \pi_\ell^{(t-1)} f(x_i|\phi_\ell^{(t-1)})}. \end{aligned} \tag{22}$$

Entrée : $\mathbf{Y}$, $K_{\max}$, T
$\{T$ est le nombre d'itérations$\}$
$\alpha^{(0)}$, $\boldsymbol{\phi}^{(0)} = (\boldsymbol{\mu}^{(0)}, \Sigma^{(0)})$, $\boldsymbol{\nu}^{(0)}$ {Initialisation}
Pour $t = 1$ to T **:**
  Étape E : calculer $\omega_{i,k}^{(t-1)}$, selon eq. (22),
    pour tout $i = 1 \dots N$, $k = 1, \dots K_{\max}$,
  Étape M : mettre à jour les paramètres
    $\boldsymbol{\nu}^{(t)} = (\nu_1^{(t)}, \dots \nu_{K_{\max}-1}^{(t)})$ selon eq. (20)
    $\boldsymbol{\pi}^{(t)} = (\pi_1^{(t)}, \dots \pi_{K_{\max}}^{(t)})$ selon eq. (11)
    $\boldsymbol{\mu}^{(t)} = (\mu_1^{(t)}, \dots \mu_{K_{\max}}^{(t)})$ selon eq. (18)
    $\boldsymbol{\Sigma}^{(t)} = (\Sigma_1^{(t)}, \dots \Sigma_{K_{\max}}^{(t)})$ selon eq. (19)
    $\alpha^{(t)}$ selon eq. (21)
Fin pour
Sortie : $\alpha^{(T)}$, $\boldsymbol{\phi}^{(T)} = (\boldsymbol{\mu}^{(T)}, \boldsymbol{\Sigma}^{(T)})$, $\boldsymbol{\nu}^{(T)}$

Algorithm 1: Algorithme TDPM-EM proposé.

Remarquons que dans notre cadre, nous ne sommes pas concernés par le changement d'assignation (*label-switching*) étant donné la représentation tronquée du DP et la nature déterministe des mises à jour dans l'algorithme EM. L'algorithme 1 résume la procédure proposée. Enfin pour chaque observation $\mathbf{y}_i$, la variable d'assignement z_i est prédite à l'aide du MAP conditionnellement au paramètre estimé $(\alpha^{(T)}, \boldsymbol{\phi}^{(T)}, \boldsymbol{\nu}^{(T)})$, *i.e.*,

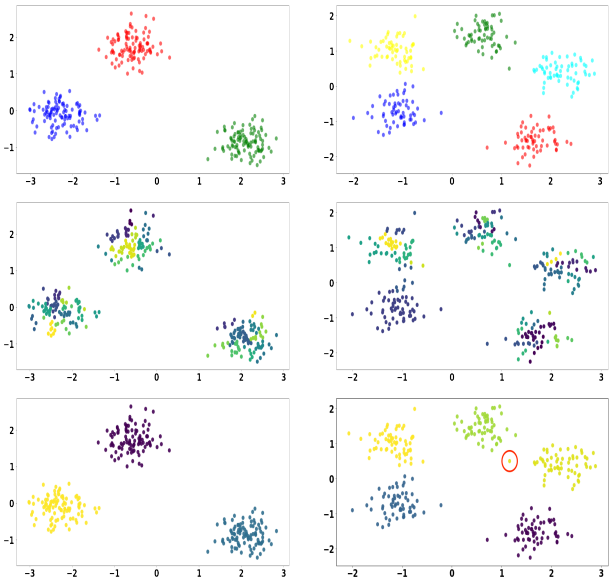
$$\hat{z}_i = \underset{k \in [\![1, K_{\max}]\!]}{\arg\max} \ \omega_{i,k}^{(T)}, \qquad i = 1 \dots N. \tag{23}$$

5 Résultats numériques

Cette partie illustre la pertinence de l'algorithme proposé TDPM-EM sur un jeu de données simulé et sur deux tâches de segmentation d'images. Les expériences sont effectuées sur un ordinateur portable personnel avec une implémentation en Python.

5.1 Exemple sur les données simulées

Dans une première expérience, nous préparons deux ensembles de données synthétiques de $N = 300$ échantillons en dimension $d = 2$ générées à partir du modèle de mélanges des gaussiennes, avec respectivement 3 et 5 classes. La valeur de $K_{\max}$ est fixée à 100 et α est initialisé à 1.2. La figure 1 montre les données simulées (première ligne), les résultats de *clustering* après 200 itérations lors de la mise à jour de α (deuxième ligne) et sans la mise à jour de α (troisième ligne). Dans les deux cas, nous obtenons de meilleurs résultats en mettant à jour le paramètre α. Plus particulièrement, pour les deux expériences, un seul point est mal classé lors de la mise à jour de α. Bien qu'aucune comparaison quantitative n'est donnée, on voit clairement au travers



(a) (b)

FIGURE 1 – *Clustering* sur des données synthétiques pour (a) 3 classes, (b) 5 classes. De haut en bas : les données simulées, les résultats de *clustering* sans la mise à jour de α ($\alpha = 1.2$), avec la mise à jour de α ($\alpha^{(0)} = 1.2$). Le cercle rouge indique le point mal classé lors de la mise à jour de α.

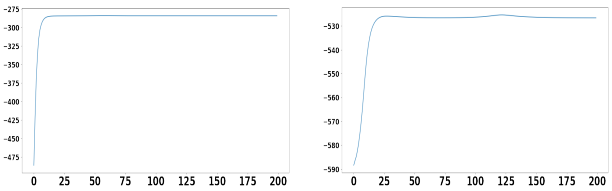


FIGURE 2 – Évolution de la log-vraisemblance au cours de 200 itérations de TDPM-EM avec la mise à jour de α : (À gauche) 3 classes, (À droite) 5 classes.

de ces simples exemples l'intérêt de la mise à jour du paramètre α. La figure 2 illustre la log-vraisemblance au cours des itérations lors de la mise à jour de α. Le log-vraisemblance converge assez rapidement après 25 itérations.

5.2 Segmentation d'image

Dans la suite, les résultats de TDPM-EM sont obtenus en mettant à jour α et après 300 itérations.

Évaluation de la robustesse au bruit. On évalue maintenant la performance de l'algorithme TDPM-EM en segmentant une image *phantom* de taille 512×512, soit $N = 262\,144$ pixels. La valeur de $K_{\max}$ est fixée

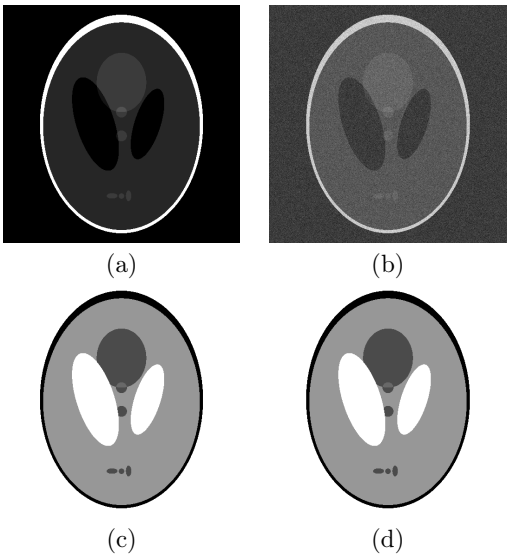


FIGURE 3 – Illustration des résultats de la segmentation sur l'image *phantom* obtenus en utilisant TDPM-EM ; (**a**) image originale, (**b**) image bruitée (σ=25, PSNR=20.17dB), (**c**) image segmentée à partir de (a), (**d**) image segmentée à partir de (b).

à 100. Dans cette expérience, la segmentation est effectuée sur l'image d'origine et sur l'image bruitée par un bruit gaussien additif d'écart-type $\sigma = 25$ (correspondant à un PSNR=20.17 dB). La figure 3 montre les résultats de la segmentation pour les deux configurations. Dans les deux cas, on voit clairement que la segmentation est précise et similaire. Cette observation est en faveur d'une forme de robustesse au bruit.

Évaluation différentes niveaux de troncature. Dans cette expérience, on utilise une image numérique du cerveau disponible sur la base BrainWeb [1] [20]. L'image utilisée est de taille 451×539, soit $N = 243\,089$ pixels. L'objectif est de distinguer les tissus cérébraux, à savoir les trois classes suivantes : la substance blanche (SB), la substance grise (SG) et le liquide cérébrospinal (LCS). Une étape de pré-traitement est souvent appliquée pour retirer les tissus non cérébraux (*e.g.* le crâne) avant d'appliquer la segmentation [3] sur les pixels liés aux tissus cérébraux. Dans cette expérience, nous segmentons tout d'abord l'image complète. Trois réalisations différentes utilisant TDPM-EM avec $K_{\max} = 50$, 100 et 150 ont été effectuées. Ensuite, nous faisons un post-traitement sur les résultats en extrayant les positions associées aux pixels des tissus non cérébral. La figure 4 illustre les résultats de ces trois réalisations. Nous remarquons que les trois niveaux de troncature fournissent des résultats similaires. Dans les trois cas, après le post-traitement, 3 classes sont obtenues sur les tissus cérébraux qui correspondent aux SB, SG, LCS. En conclusion préliminaire, on peut en déduire que le niveau de troncature a un impact marginal sur les performances de TDPM-EM lors de la mise à jour de α.

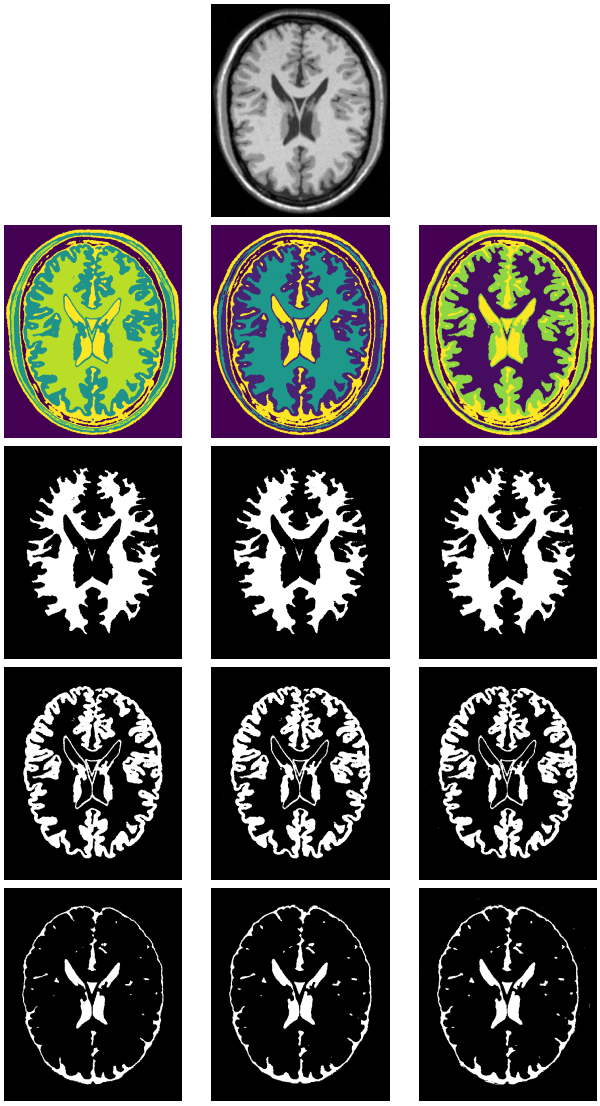


FIGURE 4 – Comparaison de TDPM-EM selon trois niveaux $K_{\max}$ de troncature. En haut : Image IRM originale issue de BrainWeb. Ci-dessous : (Colonne de Gauche) $K_{\max} = 50$, (Centre) $K_{\max} = 100$, (Droite) $K_{\max} = 150$. De haut en bas sont l'image segmentée complète et les composantes associées aux types de tissus correspondants aux SB, SG, LCS.

1. http://brainweb.bic.mni.mcgill.ca/brainweb/

6 Conclusion

Cet article propose une approche pour calculer le MAP pour le modèle DPM en utilisant la représentation *stick-breaking* tronquée. Cette approche inclut l'apprentissage du paramètre de concentration α. Nous adoptons ici un algorithme EM qui permet de mettre à jour alternativement les paramètres des composantes, les variables latentes et le paramètre α. Les résultats sur les données simulées et sur la segmentation d'image montrent que l'apprentissage du paramètre α améliore considérablement la sélection du nombre de composantes (classes) dans un modèle DPM pour le problème de *clustering*. Les travaux en cours visent à généraliser l'approche pour un modèle plus souple, tel que les mélanges par processus de Pitman-Yor, connus pour modéliser plus finement le nombre de classes.

Références

[1] E. Hancer and D. Karaboga, "A comprehensive survey of traditional, merge-split and evolutionary approaches proposed for determination of cluster number," *Swarm and Evolutionary Computation*, 2017.

[2] J. Han, M. Kamber, and J. Pei, *Data mining : Concepts and techniques.* Morgan Kaufmann, 2012.

[3] A. R. F. da Silva, "Bayesian mixture models of variable dimension for image segmentation," *Computer Methods and Programs in Biomedicine*, 2009.

[4] D.-S. Lee, "Effective gaussian mixture learning for video background subtraction," *IEEE Trans. Patt. Anal. Mach. Intell.*, 2005.

[5] J. Puzicha, T. Hofmann, and J. M. Buhmann, "Histogram clustering for unsupervised segmentation and image retrieval," *Pattern Recognition Letters*, 1999.

[6] C. Samson, L. Blanc-Feraud, G. Aubert, and J. Zerubia, "A variational model for image classification and restoration," *IEEE Trans. Patt. Anal. Mach. Intell.*, 2000.

[7] L. Hermes, T. Zöller, and J. M. Buhmann, "Parametric distributional clustering for image segmentation," in *Computer Vision — ECCV.* Springer Berlin Heidelberg, 2002.

[8] A. K. Jain, "Data clustering : 50 years beyond k-means," *Pattern recognition letters*, 2010.

[9] F. Murtagh, A. E. Raftery, and J.-L. Starck, "Bayesian inference for multiband image segmentation via model-based cluster trees," *Image and Vision Computing*, 2005.

[10] C. Antoniak, "Mixtures of dirichlet processes with applications to bayesian nonparametric problems." *The annals of statistics*, 1974.

[11] P. Orbanz and J. M. Buhmann, "Nonparametric bayesian image segmentation," *International Journal of Computer Vision*, 2008.

[12] J. Sodjo, A. Giremus, F. Caron, J. Giovannelli, and N. Dobigeon, "Joint segmentation of multiple images with shared classes : A bayesian nonparametrics approach," in *Proc. IEEE Workshop Stat. Signal Process.*, 2016.

[13] M. D. Escobar and M. West, "Bayesian density estimation and inference using mixtures," *J. Amer. Stat. Assoc.*, 1995.

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society : Series B (Methodological)*, vol. 39, no. 1, 1977.

[15] T. Ferguson, "A bayesian analysis of some nonparametric problems," *The Annals of Statistics*, 1973.

[16] J. Sethuraman, "A constructive definition of dirichlet priors," *Statistica Sinica*, 1994.

[17] D. M. Blei and M. I. Jordan, "Variational inference for dirichlet process mixtures," *Bayesian Analysis*, 2006.

[18] H. Ishwaran and L. James, "Gibbs sampling methods for stick-breaking prior," *J. Amer. Stat. Assoc.*, 2001.

[19] Y. W. Teh, "Dirichlet processes," in *Encyclopedia of Machine Learning.* Springer, 2010.

[20] C. A. Cocosco, V. Kollokian, R. K.-S. Kwan, G. B. Pike, and A. C. Evans, "Brainweb : Online interface to a 3d mri simulated brain database," *NeuroImage*, vol. 5, p. 425, 1997.

Détection d'émotions sur données multimodales restreintes par réseaux triples

Kergann Le Cornec et Vincent Barra

Université Clermont Auvergne, CNRS, LIMOS, Clermont-Ferrand, F63000, France

Résumé

Nous présentons dans cet article un modèle d'apprentissage profond permettant une **reconnaissance d'émotions multimodales**, combinant des images de visages et des enregistrements sonores de phrases codés par des coefficients cepstraux. L'approche utilise un **réseau triple** afin d'augmenter la taille de la base d'apprentissage et de représenter les différentes émotions dans un espace vectoriel.

Le modèle, basé sur le réseau Facenet [FS15], permet de placer correctement les triplets de test avec plus de 85% de précision en utilisant seulement 200 couples de type image-son pour chaque émotion étudiée. Ainsi que classifier avec plus de 80% de précision les couples Image-Son. La méthode développée permet également de séparer correctement ces émotions, sans indiquer implicitement le label de la donnée apprise au réseau, ni le nombre de classes totales.

Mots-clef : Reconnaissance d'émotion, Réseau Triple, Multimodale, Image, Analyse cepstrale.

1 Introduction

La reconnaissance d'émotions faciales et vocales est indispensable dans la communication. Pendant une conversation, l'auditeur va détecter les émotions du locuteur et pourra répondre en fonction. Pour ce faire, il peut par exemple utiliser le ton de la voix, la gestuelle ou les expressions du visage. C'est aussi une fonction essentielle pendant les interactions humain-machine, une meilleure compréhension des émotions menant à une meilleure compréhension de l'utilisateur et donc à une amélioration des décisions prises par l'intelligence artificielle.

Les émotions sont exprimées de différentes manières, par nos gestes, nos expressions, notre activité cérébrale, l'intensité de notre voix, etc. Nous utilisons dans la suite deux types de données : des images représentant des expressions faciales et des enregistrements sonores correspondant à différentes émotions. Combiner ces informations multimodales et complémentaires doit permettre d'obtenir un modèle performant de détection d'émotions.

De nos jours, les méthodes d'apprentissage profond sont très performantes dans les domaines de la reconnaissance d'images et de sons. Cependant ces modèles nécessitent souvent une grande quantité de données afin de pouvoir correctement généraliser. Au contraire, les humains sont exceptionnellement doués pour classifier avec peu de données. Ils peuvent facilement apprendre une nouvelle notion avec peu d'exemples, et même reconnaître différentes formes de la même notion [LST15].

Utiliser le paradigme de l'apprentissage profond dans un contexte de très faible supervision est un challenge. Dans ce cadre, les réseaux triples [WGDM+14] sont des modèles très intéressants car ils ne nécessitent que peu de données.

Dans cet article, nous nous intéressons au fonctionnement de ces réseaux et comment ils réagissent quand nous combinons les informations provenant des sons et des images. Le but est, avec peu de données, de classifier un couple contenant une image et un son. Nous nous limitons aux six émotions de base, décrites par le psychanalyste Paul Eckman [Ekm72] : la peur, le dégoût, la colère, la surprise, le bonheur et la tristesse. Ces émotions sont des émotions universelles présentes dans toutes les cultures.

Nous présentons tout d'abord l'état de l'art sur la détection d'émotions, puis nous décrivons les bases de données utilisées. La troisième partie détaille le modèle utilisé, son architecture et son entraînement. Nous exposons enfin et discutons les résultats obtenus par cette méthode.

Contributions. Nos contributions sont les suivantes :

— Un modèle triple convolutif pour des données multimodales restreintes.

— Une nouvelle fonction d'erreur basée sur le *Triplet Loss* [FS15].

— Une représentation vectorielle des émotions.

2 État de l'art

Ces dernières années les réseaux profonds ont été optimisés pour la reconnaissance d'images et montrent les meilleurs résultats dans ce domaine. Dans cette partie nous présentons brièvement des réseaux d'apprentissage profond pour la reconnaissance ou la représentation d'émotions.

2.1 Représentation et reconnaissance sur une base de données restreinte

Rao et al. présentent dans [RWC16] une représentation des six émotions de base. Les auteurs utilisent un réseau siamois basé sur Alexnet [KSH12], et entraînent leur modèle sur la base de données Nimstim [Tea09]. La comparaison avec notre modèle est difficile car les auteurs ne donnent pas de métrique de précision, mais seulement la représentation vectorielle des émotions.

Dans [Kea15], les auteurs présentent un réseau profond entraîné sur la base audio-visuelle de données EmotiW 2015. En combinant un réseau convolutif profond pour les images, un réseau *deep belief* pour les sons, un auto-encodeur pour apprendre les caractéristiques spatio-temporelles et un réseau peu profond pour extraire les caractéristiques des mouvements des lèvres,les auteurs obtiennent 55.6% de précision sur les six émotions de bases.

Kim et al. [KLRL15] proposent un modèle de convolution appris sur la base de données SFEW 2.0 [DGLG11], et atteignent 61.6% de précision sur 7 classes.

Hasani et Mahoor [HM17] introduisent un réseau convolutif 3D composé de couches Inception-Resnet[SIVA16] suivies d'un LSTM permettant en utilisant des vidéos et des points d'intérêt des images d'atteindre, sur la base MMI[Mea05] (11500 images), 77.50% sur 6 émotions et sur la base Fera[BK10] (7000 images) 77.42% pour 5 émotions.

Vemulapali et Agarwala utilisent dans [VAA18] un réseau triple, combinant pour *l'embedding*, les réseaux Facenet [FS15] et DenseNet [HLvdM17]. Les auteurs introduisent une nouvelle fonction d'erreur et obtiennent une précision de 88.6% sur 8 émotions. Le réseau est entraîné sur la base de données [MHM17] de 150K images différentes ce qui est peu pour entraîner un réseau totalement, sans technique de *finetuning*. La base de données est séparée en trois types de triplets : ceux contenant trois images d'une classe, ceux contenant trois images de deux classes et ceux contenant trois images de trois classes.

2.2 Représentation et reconnaissance sur une base de données large

Ces techniques ne sont pas comparables à notre approche, mais leur étude permet notamment de positionner notre approche en termes de performances.

Harár et al. utilisent un réseau profond convolutif [HBD17] sur des données audio brutes provenant de la base Berlin Database of Emotional Speech [BPR+05]. Après les couches de convolution, les auteurs appliquent une couche d'agrégation max d'une part, et une couche d'agrégation moyenne d'autre part, puis concatènent les deux vecteurs calculés avant d'envoyer le vecteur final à la couche complètement connectée pour classification, avec une précision de 96.97% sur trois classes.

Zhanpeng et al. [ZLLt17] présentent un réseau convolutif profond ayant appris sur la large base de données d'images faciales *The Extended CohnKanade* [PCK+10], et atteignent 98.5% de précision.

Niu et al. présentent [NZN+17] une manière d'augmenter la base de données de phrases sonores IEMOCAP [el.08]. Ils se basent sur le principe de symétrie des lentilles convexes, afin de créer des spectrogrammes de sons plausibles à partir de spectrogrammes déjà existants. Les auteurs utilisent par la suite un réseau profond convolutif pour arriver à une précision de 99.25%.

Hossain et Muhammad [HM19] construisent un modèle convolutif multimodal sur des vidéos, appris sur des grandes bases de données et atteignant 99.9% de précision sur trois émotions. Les auteurs utilisent un réseau convolutif 2D sur le spectrogramme Mel conventionnel des sons, un réseau convolutif 3D sur la vidéo, puis une couche *Extreme learning machine* sur les caractéristiques trouvées.

Nous pensons que beaucoup de travail reste à faire dans le domaine de l'apprentissage sur données restreintes. C'est pour cette raison que nous nous limitons à une base de données restreinte de sons et d'images décrite dans la section 3. Nous combinons les informations présentes dans ces deux types de données afin de

créer un modèle performant. Nous avons fait le choix des réseaux triples décrits dans la section 4.1 pour leurs performances d'apprentissage sur données restreintes.

3 Bases de données

Nous présentons dans cette section les bases de données d'images faciales et de phrases sonores utilisées.

3.1 Image-NimStim

Pour les images faciales, nous utilisons la base de données NimStim, présentée dans [Tea09] et détaillée dans le tableau 1. Dans ce tableau, nous indiquons également le nombre de triplets utilisés par le réseau décrit dans la section 4.1.

Les données sont des images RGB de taille 506×650, un exemple est donné figure 3.

3.2 Son-Ravdess

Pour les sons, nous utilisons la base de données Ravdess détaillée dans le tableau 2, et introduite dans [SF18]. Ce sont des phrases de trois secondes enregistrées en studio, chacune prononcée six fois avec les intonations correspondantes aux différentes émotions de base. Dans le tableau 2, nous indiquons également le nombre de triplets utilisés par le réseau décrit dans la section 4.1.

Les données temporelles sont traitées par analyse cepstrale, afin de fournir un ensemble de caractéristiques pertinentes en entrée du réseau profond. Plus précisément, les coefficients cepstraux de fréquence Mel (MFCC), introduits dans [DM80], sont utilisés dans la suite pour chaque enregistrement. Le calcul des coefficients MFCC est expliqué schématiquement sur la figure 1, et détaillé ci après.

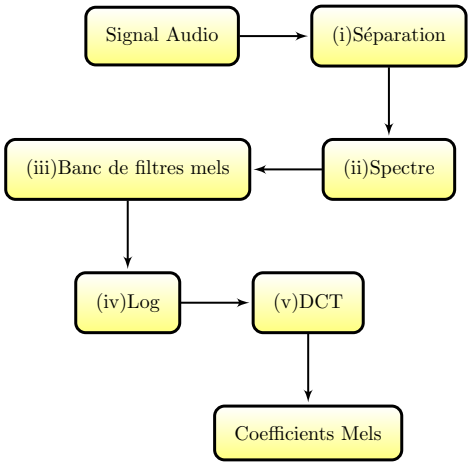


FIGURE 1 – Principe des MFCC.

(i) Un fenêtrage du signal temporel sur de petites fenêtres de temps (10-20 ms) est effectué ;

(ii) Le spectrogramme en puissance pour chaque fenêtre est calculé. Nous savons que l'oreille réagit différemment selon différentes fréquences et indique au cerveau les fréquences présentes dans le son. Notre réseau utilisera cette information fréquentielle pour détecter les émotions présentes ;

(iii) Le spectrogramme est ensuite pondéré par un banc de filtres triangulaires espacés selon l' **échelle de Mel** [SVN37]. Le ton est l'une des caractéristiques principales de la parole, nous pouvons la mesurer en fréquence. L'échelle de Mel est une échelle qui copie ce que l'humain perçoit. Elle adapte la mesure en fréquence du ton pour être plus proche de ce que notre oreille entend. Nous identifions mieux les petits changements dans les basses fréquences que les petits changements dans les hautes fréquences.

Nous appliquons un banc de filtres mel aux signaux, qui permettent d'extraire des bandes de fréquences du spectrogramme et de regrouper les données importantes. Les filtres sont schématisés sur la figure 2 [Mea13] où l'on remarque le plus grand nombre de filtres dans les basses fréquences que dans les hautes fréquences. La conversion d'une fréquence f (en Hz) à l'échelle Mel se fait selon la formule $m = 2595 * \log(1 + f/700)$. Pour chaque filtre, l'énergie totale est calculée par sommation, et donne une information sur la répartition énergétique dans chaque région fréquentielle de l'échelle Mel. Le premier filtre donne par exemple l'indication de l'énergie autour de la fréquence nulle ;

(iv) Un opérateur logarithme est appliqué pour amplifier les petites différences dans les faibles énergies ;

(v) Une transformée en cosinus discret (DCT) est enfin effectuée. En effet, à cette dernière étape, les coefficients mels sont corrélés puisque les filtres se recouvrent partiellement, La DCT permet de décorréler les coefficients calculés, pour aboutir aux MFCC. De manière usuelle, en reconnaissance vocale, seuls les coefficients cepstraux de 2 à 13 sont conservés, les autres coefficients représentant des changements trop rapides et ne contribuant

Émotion	Colère	Dégoût	Peur	Bonheur	Tristesse	Surprise
Images d'entraînement	63	70	68	113	68	25
Images de test	10	10	10	10	10	10
Triplets d'entraînement	671 832	813 855	772 242	1 860 432	772 242	114 600
Triplets de test	2,25K	2,25K	2,25K	2,25K	2,25K	2,25K

Table 1 – Base de données faciale : NimStim.

	Émotions
Sons d'entraînement	100
Sons de test	10
Triplets d'entraînement	2,475M
Triplets de tests	2,25K

Table 2 – Base de donnée RAVDESS représentant du nombre de phrases sonores en fonction de l'émotion (colère, dégoût, peur, bonheur, tristesse, surprise).

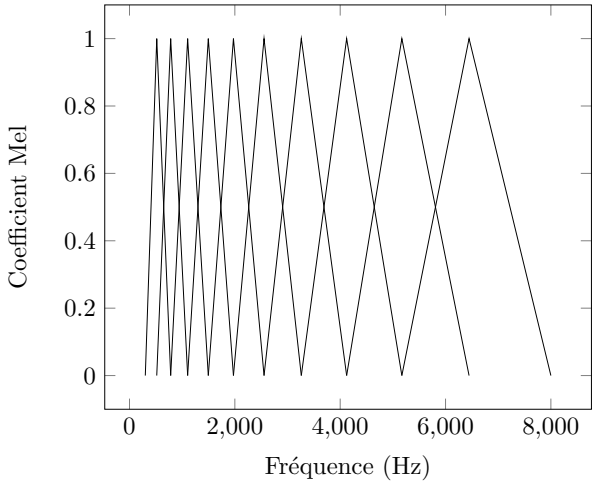


Figure 2 – Exemple des 10 premiers filtres mels

pas à la reconnaissance vocale.

La figure 3 présente un exemple de spectrogramme MFCC calculé sur une phrase de la base de données.

Chaque ligne verticale représente 10 ms de la phrase choisie. Les 13 lignes (0-12) représentent les coefficients MFCC. Pour la représentation de l'énergie, nous avons utilisé une représentation *Cool-warm*. La couleur représente l'énergie contenue dans chaque coefficient mel, du bleu au rouge par niveau croissant d'énergie.

3.3 Couples

Nous avons décidé d'utiliser des couples **Image-Son**, notés (I, S) et de nous limiter à 200 couples par classe.

	Émotion
Couples (Image-Son) d'entraînement	200
Couples (Image-Son) de test	10
Triplets de couple d'entraînement	19,9M
Triplets de couples de test	2,25K

Table 3 – Base de donnée représentant le nombre de couples Image-Son en fonction de l'émotion (colère, dégoût, peur, bonheur, tristesse, surprise).

Nous concaténons l'image de la personne et le spectrogramme du son dans une seule image. Un exemple est donné dans la figure 3.

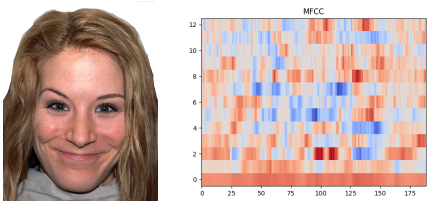


Figure 3 – Gauche : image faciale labellisée heureuse. Droite : MFCC d'une donnée labellisée heureuse de la base de sons ; abscisse : temps (ms), ordonnée : coefficents cepstraux

4 Approche

Afin de contrer le nombre de données restreintes, nous avons décidé d'adapter à notre problème un réseau triple défini par Wang et al. [WGDM+14]. Dans cette section, nous expliquons le fonctionnement du réseau et l'erreur utilisée.

4.1 Réseau Triple

Le réseau triple est schématisé sur la figure 4. Il utilise le réseau profond **Inception-Resnet-v2** [SIVA16] en lieu et place du réseau *Inception* utilisé dans [FS15], car il a été montré que le réseau *Inception-Resnet-v2* apprenait mieux et plus rapidement avec des connexions résiduelles. Le principe est de donner au réseau un triplet d'instances au lieu d'une donnée seule. Le triplet est composé de trois données :

— un couple ancre A composé d'une image et d'un son de la même classe, noté $C_a = (I_a, S_a)$;

— un couple positif P de la même classe que l'ancre, noté $C_p = (I_p, S_p)$;

— un couple négatif N d'une classe différente de celle de l'ancre $C_n = (I_n, S_n)$.

Nous avons calculé le nombre de triplets d'entraînement et de test dans le tableau 3 pour connaître le nombre réel de données d'apprentissage.

Le réseau va représenter ces trois instances puis modifier ses poids pour faire en sorte d'éloigner le négatif de l'ancre, et de rapprocher le positif de l'ancre. Comme dans tout processus d'apprentissage d'un réseau de neurones, ceci est réalisé en minimisant une fonction de coût, et celle utilisée ici est le triplet loss (paragraphe 4.2).

La seule information donnée au réseau est que les deux premiers couples sont de la même classe et le troisième d'une classe différente, nous notons cette information **Positif/Négatif**.

4.2 Fonction d'erreur

Dans cette partie, nous décrivons la fonction d'erreur *triplet loss* combinée au *triplet mining* [FS15]. Nous soulignons les limites de cette approche et proposons une nouvelle fonction d'erreur.

Triplet Loss

Le *Triplet Loss* est défini comme :

$$Tl(E_a, E_p, E_n) = \max(d(E_a, E_p) - d(E_a, E_n) + m, 0) \ ,$$

où E_a est la représentation (*embedding*) de l'ancre après le réseau, et E_p (respectivement E_n) est la représentation de la donnée positive (resp. négative). m est une marge et d est une métrique. Dans la suite, nous utilisons la norme euclidienne $|| \ ||_2$.

La fonction qui sera minimisée sera alors :

$$\sum_{i=1}^{N} \left[\|e_i^a - e_i^p\|_2^2 - \|e_i^a - e_i^n\|_2^2 + m \right] \ ,$$

où e^a (resp e^p, e^n) sont les coordonnées du vecteur *embedding* ancre (resp. positif, négatif) et N la dimension de l'espace de représentation. Cette fonction entraîne le réseau sur un triplet d'instances au lieu d'une instance à la fois, et permet de :

— Rapprocher deux données de la même classe, cela revient à diminuer la distance entre une ancre et une donnée positive $d(E_a, E_p)$.

— Éloigner deux données de classes différentes, cela revient à augmenter la distance entre l'ancre et une donnée négative $d(E_a, E_n)$).

Pour que les classes soient bien séparées, une marge m est ajoutée. Pour un triplet (C_a, C_p, C_n), la distance $d(E_a, E_n)$ doit être supérieure à $d(E_a, E_p)$ plus une marge. En minimisant cette fonction de coût, la distance $d(E_a, E_p)$ sera donc diminuée et la distance $d(E_a, E_n)$ sera augmentée.

Triplet Mining

Si nous choisissons la métrique précédente, le choix des triplets est capital pour la bonne convergence du modèle. Il existe trois catégories de triplets :

— triplets simples : ceux dont la valeur du *triplet loss* est nulle car $d(a, p) + m < d(a, n)$;

— triplets complexes : ceux où le négatif est plus proche de l'ancre que le positif $d(a, n) < d(a, p)$;

— triplets semi-complexes : donnant une valeur du *triplet loss* positive $d(a, p) < d(a, n) < d(a, p) + m$.

Les différents triplets sont illustrés dans la figure 5.

Nous allons chosir, pendant l'entraînement, les triplets utiles à l'entraînement. Avec la métrique originale utilisée dans [FS15], les triplets simples ne sont pas utiles pour l'entraînement, nous allons donc seulement choisir les triplets complexes et semi-complexes.

Fonction d'erreur utilisée

Tous les triplets ne donnent pas des informations utiles (en particulier les triplets simples). De plus, le choix de la marge m change en fonction du type de données. Fort de ces constats, nous avons ajouté une couche sigmoïde à la fin de notre réseau *Inception-Resnet-v2*. Chaque valeur contenue dans le vecteur final sera donc entre 0 et 1. Si nous notons N la dimension de ce vecteur, sa norme euclidienne sera au maximum

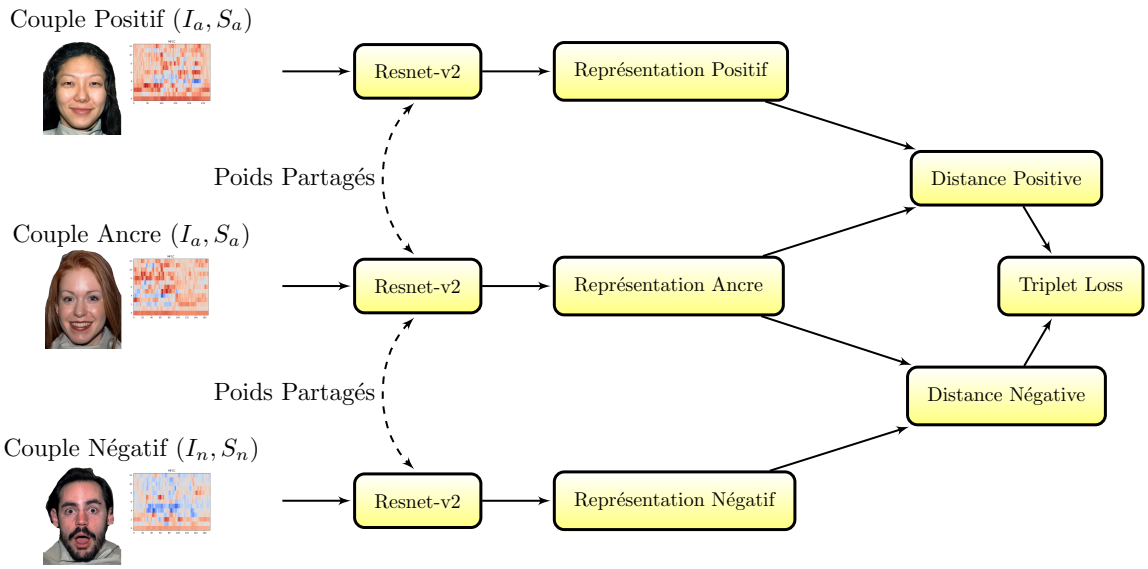


FIGURE 4 – Principe du réseau triple.

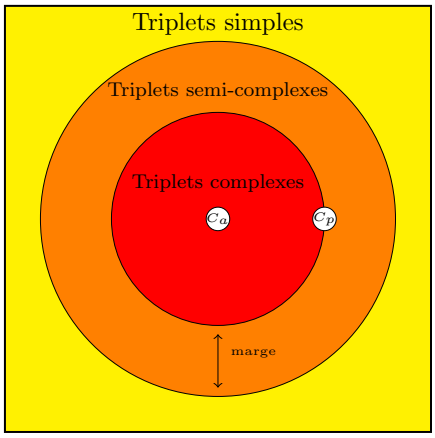


FIGURE 5 – Types de triplets.

$\sqrt{N}$. Nous pouvons alors transformer la fonction d'erreur en :

$$Loss(E_a, E_p, E_n) = d(E_a, E_p) - d(E_a, E_n) + \sqrt{N}$$

ou encore

$$Loss(E_a, E_p, E_n) = \sum_{i=1}^{N} \left[(e_i^a - e_i^p)^2 - (e_i^a - e_i^n)^2 + \sqrt{N} \right] .$$

Cette fonction assure que l'erreur ne tombera jamais à 0 et que tous les triplets seront utiles à l'entraînement. Pour rendre cette fonction non-linéaire, être sensibles aux petites erreurs et améliorer l'entraînement, nous appliquons une fonction log sur chaque distance.

$$LLoss(E_a, E_p, E_n) = \log(d(E_a, E_p)) - \log(d(E_a, E_n)) + \log(\sqrt{N}) .$$

5 Entraînement

Le réseau *Inception-Resnet-v2* [SIVA16] a été entraîné sur des visages avec un objectif de reconnaissance faciale, et nous avons utilisé les poids de ce réseau entraîné pour l'initialisation de notre méthode. Il est très difficile d'entraîner correctement les poids des premières couches de convolution sur des données restreintes. Cette méthode de *finetuning* permet d'obtenir des poids acceptables sur les premières couches de convolution et d'améliorer la performance du réseau. Nous avons continué l'entraînement sur toutes les couches afin d'améliorer les poids pour correspondre à nos données.

Par souci de mémoire, nous avons sélectionné aléatoirement pour chaque epoch 20 images dans chaque classe et appris sur tous les triplets possibles. Une epoch représente donc tous les triplets possibles d'un sous groupe de la base de données entière.

Le réseau a été entraîné avec l'optimiseur Adam avec un *learning rate* à 0.001 qui décroît exponentiel-

lement toutes les epochs, 40% de dropout est utilisé sur les dernières couches complètement connectées afin d'éviter le sur apprentissage.

6 Résultats

Le réseau à été implémenté à l'aide du *framework* Tensorflow [1]. Pour la phase d'entraînement, une carte graphique TitanX Pascal ainsi que 64 GiB de RAM ont été utilisés.

Dans cette section nous présentons la précision du modèle, l'erreur et la représentation vectorielle des six émotions de base pendant apprentissage.

6.1 Précision

Nous présentons, dans le tableau 4, les résultats de notre approche pour différents pré-traitements et différents réseaux triples. Dans ce tableau, la précision est calculée sur le nombre de triplets que le réseau place correctement i.e si $d(A, P) < d(A, N)$.

Afin de comparer nos résultats sur cette métrique, nous avons aussi entraîné notre réseau sur les images seules, et les sons seuls. Nous avons également entraîné le réseau sur le son pur et non le spectrogramme, puis utilisé d'autres réseaux pour l'embedding. Les résultats sont meilleurs avec les spectrogrammes, le réseau est de plus mieux adapté à des images et identifie mieux les régions significatives dans les images. Les tests ont été effectués sur la base décrite dans la section 3.3. Les personnes (images faciales et phrases sonores) testées n'ont jamais été vues pendant la phase d'entraînement. Dans le tableau, la colonne *Temps* décrit le temps total d'entraînement du réseau. Notre modèle permet de placer correctement les triplets de test à une précision de **85%**.

Afin de comparer nos résultats à l'état de l'art, nous avons chosi de calculer la précision sur le nombre de couples bien classifiés. Nous avons calculé la distance entre le couple testé et des couples provenant de la base d'entraînement (10 couples pour chaque classe). Nous additionnons par la suite les distances pour chaque classe et le minimum est la classification finale trouvée. Nous arrivons à une précision de plus de 80%. Nous remarquons sur la matrice de confusion (tableau 5), ainsi que sur la représentation des couples tests représentée sur l'image (f) de la figure 7, que le réseau n'a pas correctement appris la tristesse. Elle se confond beaucoup avec la peur et la surprise. La colère aussi se confond aussi avec le dégout, les deux groupes sont très proches.

1. https ://tensorflow.org

Nous avons comparé nos résultats à l'état de l'art dans le tableau 6 en fonction du nombre de données utilisées. Nous pouvons remarquer que les réseaux triples sont plus performants que les réseaux classiques quand le nombre d'instances est limitée.

6.2 Représentation des émotions

Pour avoir une représentation vectorielle à deux dimensions, nous rajoutons une couche entièrement connectée (FC) de taille deux à la fin du réseau *Inception-Resnet-v2*. Nous remarquons sur la figure 7 que le réseau apprend correctement et offre une représentation des émotions de base. Il place la plupart des instances dans les bonnes régions et arrive à séparer les différentes classes sans indication implicite du nombre de classes ou du label de la donnée, uniquement avec l'indication *Positif/Négatif*. Nous pouvons noter que pendant la phase d'entraînement, le modèle confond la peur et la surprise, ainsi que la colère et le dégoût. Le réseau met un certain temps à bien apprendre la différence entre ces classes car elles sont assez similaires.

7 Conclusions et perspectives

Nous avons présenté un modèle de classification d'émotions utilisant des informations sonores et visuelles d'une base de données restreintes. Nous avons utilisé un réseau triple basé sur le réseau *Inception-Resnet-v2*. Pour l'entraînement, seule l'information Positif/Négatif est disponible. Nous ne donnons jamais au réseau l'information de classe de l'instance ni le nombre total de classes. Le réseau classifie correctement les triplets à plus de 85% et regroupe de manière pertinente les différentes classes dans différents groupes. Il classifie aussi les couples Image-Son, sur 6 émotions différentes, à plus de 80% de précision. *L'embedding* appris est correct et utilisable pour la classification d'émotions.
Nous envisageons maintenant d'autres méthodes de pré-traitement des données, ou une refonte de l'architecture du réseau *embedding* pour apprendre seulement sur des données restreintes, sans utiliser de méthodes de *finetuning*.

Références

[BK10] T. Bänziger and K.R.Scherer. Introducing the geneva multimodal emotion portayal (gemep) corpus. 2010.

Nombre de données	Réseau	Traitement	Précision	Temps
Image				
407	Triplet Alexnet	RGB	72%	12h
407	Triplet Resnet	RGB	75%	16h
407	Triplet *Inc-Res*	RGB	80%	16h
Audio				
600	Triplet *Inc-Res*	Son pure	70%	12h
600	Triplet *Inc-Res*	MFCC	75%	12h
Image + Son Pur				
1200	Triplet *Inc-Res*	RGB+Son Pure	60%	24h
1200	Triplet *Inc-Res*	RGB+filtre lissant	70%	24h
Image + Spectrogramme				
1200	*Inc-Res+Triplet Loss*	RGB+Mel-Spec	82%	24h
1200	**Linear Triplet Loss**	**RGB+Mel-Spec**	**84%**	**24h**
1200	**Non Linear Triplet Loss**	**RGB+Mel-Spec**	**85%**	**24h**

Table 4 – Comparaison de différentes approches. On note Inc-Res le réseau Inception-Resnet-v2.

Emotions	Happy	Angry	Disgusted	Surprised	Sad	Fear
Happy	997	3	0	0	0	0
Angry	10	776	214	0	0	0
Disgusted	83	47	870	0	0	0
Surprised	0	0	0	730	0	270
Sad	198	0	144	1	473	184
Fear	0	0	0	30	0	970

Table 5 – Matrice de confusion, 1000 tests par classe, 80.3% de précision

Nombre de données	Réseau	Traitement	Précision
SFEW 2.0 (958)	CNN-State [KLRL15]	Image	61.6%
FERA (7000)	3D Inc-Res[HM17]	Vidéo+Points d'intérêts	77.42%
MMI (11500)	3D Inc-Res[HM17]	Vidéo+Points d'intérêts	77.50%
Nimstim+Ravdess (1200)	**Triplet-Inc-Res**	**Image+Son Mel-Spec**	**80.3%**
CK	3D-CNN+ELM [HM19]	MFCC (Audio) + Niveau de Gris	87.5%
CK+ (450K)	AFFNet-CL [VAA18]	Image	88.6%
The Extended ChohnKanade	DCN [ZLLt17]	Niveau de Gris	98.5%

Figure 6 – Comparaison de notre modèle avec l'état de l'art.

[BPR+05] Burkhardt, F. Paesche, A. Rolfes, M. Sendlmeier, and W.F. Weiss. A database of German emotional speech. 2005.

[DGLG11] Abhinav Dhall, Roland Goecke, Simon Lucey, and Tom Gedeon. Static facial expression analysis in tough conditions : Data,evaluation protocol and benchmark. 2011.

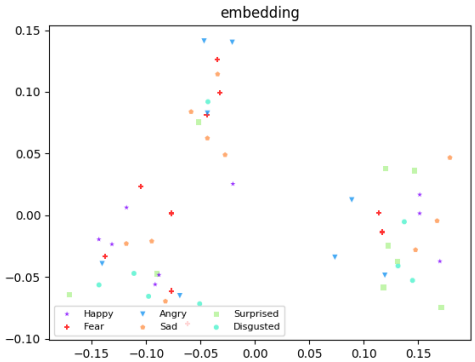
[DM80] S.B. Davis and P. Mermelstein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. 1980.

[Ekm72] Paul Ekman. Emotion In Human Face. 1972.

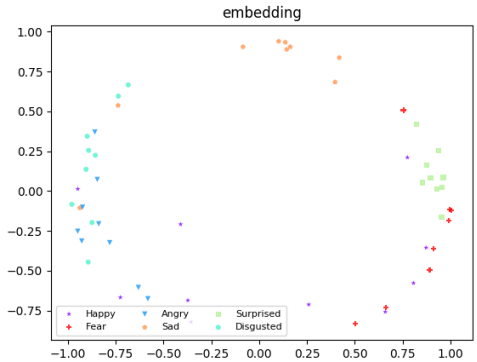
[el.08] Busso Carlos el.al. IEMOCAP : interactive emotional dyadic motion capture database. 2008.

[FS15] James Philbin Florian Schroff, Dmitry Kalenichenko. FaceNet : A Unified Embedding for Face Recognition and Clustering. 2015.

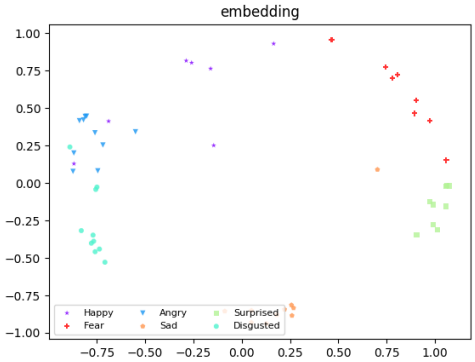
[HBD17] Pavol Harár, Radim Burget, and Malay Kishore Dutta. Speech Emotion Recogntion with Deep Learning. 2017.



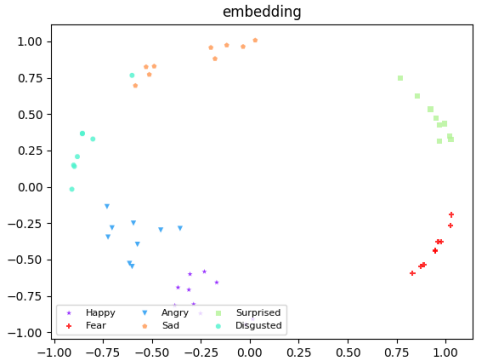
(a) Représentation Initiale.



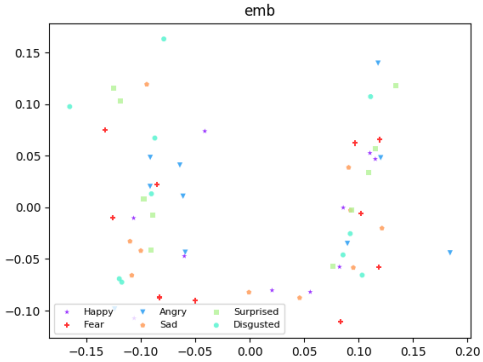
(b) Représentation après 15 Epoch.



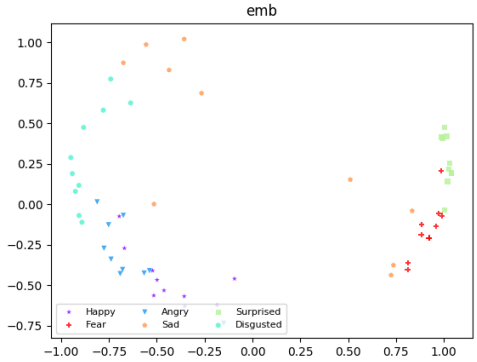
(c) Représentation après 30 Epochs.



(d) Représentation finale après 50 Epochs.



(e) Sur-apprentissage après 50 Epochs.



(f) Représentation sur des couples tests

FIGURE 7 – Représentation des émotions pendant la phase d'apprentissage et de test sur couple Image-Son.

[HLvdM17] Gao Huang, Zhuang Liu, and Laurens van der Maaten. Densely Connected Convolutional Network. 2017.

[HM17] Behzad Hasani and Mohammad H. Mahoor. Facial Expression Recognition Using Enhanced Deep 3D Convolutional Neural Networks. 2017.

[HM19] M. Shamim Hossain and Ghulam

Muhammad. Emotion recognition using deep learning approach from audio–visual emotional big data. 2019.

[Kea15] Samira Ebrahimi Kahou and Xavier Bouthillier et al. EmoNets : Multimodal deep learning approaches for emotionrecognition in video. 2015.

[KLRL15] Bo-Kyeong Kim, Hwaran Lee, Jihyeon Roh, and Soo-Young Lee. Hierarchical committee of deep cnns with exponentially-weighted decision fusion for staticfacial expression recognition. 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskeve, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Network. 2012.

[LST15] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. 2015.

[Mea05] M.Pantic and M.Valstar et al. Web-based database for facial expression analysis. 2005.

[Mea13] Yusnita M.A. and Paulraj M.P. et al. Analysis of Accent-Sensitive Words in Multi-Resolution Mel-Frequency Cepstral Coefficients for Classification of Accents in Malaysian English . 2013.

[MHM17] A. Mollahosseini, B. Hasani, and M. H. Mahoor. Affectnet : A database for facial expression, valence, and arousal computing in the wild. 2017.

[NZN+17] Yafeng Niu, Dongsheng Zou, Yadong Niu, Zhongshi He, and Hua Tan. A breakthrough in Speech emotion recognition using Deep Retinal Convolution Neural Networks. 2017.

[PCK+10] P.Lucey, J.F. Cohn, T. Kanade, J.Saragih, Z. Ambadar, and I. Matthews. The extended cohn-kanade dataset (ck+) : A completedataset for action unit and emotion-specified expression. 2010.

[RWC16] Sanjeev Jagannatha Rao, Yufei Wang, and Garrisson W Cottrell. A Deep Siamese Neural Network Learns the human-Percieved Similarity Sturcture of Facial Expressions Without Explicit Categories. 2016.

[SF18] Livingstone SR and Russo FA. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). A dynamic, multimodal set of facial and vocal expressions in North American English. 2018.

[SIVA16] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. 2016.

[SVN37] Stanley Smith Stevens, John Volkman, and Edwin B. Newman. A scale for the measurement of the psychological magnitude pitch. Journal of the Acoustical Society of America. 1937.

[Tea09] Nim Tottenham and James W. Tanaka et al. The NimStim set of facial expressions : Judgments from untrained researth participants. 2009.

[VAA18] Raviteja Vemulapalli and Google AI Aseem Agarwala. A Compact Embedding for Facial Expression Similarity. 2018.

[WGDM+14] Hanyu Wang, Jianwei Guo, Dong-MingYan, Weize Quan, and Xiaopeng Zhang. Learning 3D Keypoint Descriptors for Non-Rigid Shape Matching. 2014.

[ZLLt17] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou tang. From Facial Expression Recognition to Interpersonal RelationPrediction. 2017.

Matching Words and Knowledge Graph Entities with Meta-Embeddings

Damien Sileo[1], Camille Pradel[1], Guillermo Eschegoye[2], Anselmo Peñs[2], Arantxa Otgi[4], Jan Milan Driu[3], Mark Cielebak[3], Ander Barena[4], et Eneko Agirre[4]

[1]Synapse Développement,Toulouse
[2]Universidad Nacional de Educación a Distanca, Madrid
[3]Zurich University of Applied Scienes, Zurich
[4]IXA NLP Group, University of the Basque Country, Donostia

31 mai 2019

Résumé

Word vectors are a key component for matching distinct textual units semantically. However, they are not directly applicable for matching text with structured data, for which graph embeddings exist. In this work, we propose a flexible method in order to map the representation of graph embeddings to word embeddings representation. Thus, we can improve word embeddings with a weighted average with mapped graph embeddings. We evaluate our models on the task of matching natural language questions and SPARQL queries, and significantly improve queries matching accuracy. We also evaluate word meta-embeddings intrinsically and show improvements over previous models.

Mots-clef : Question answering, Knowledge Graphs, Word Embeddings, Graph Embeddings, Meta-Embedding

1 Introduction

Structured data has become ubiquitous, abundant and involved in numerous applications. Knowledge bases like DBpedia, Wikidata, OpenCyc [FEMR15] provide large and growing structured resources. They contain millions of facts represented as triplets such as (Paris, LOCATED_IN, France). Formal languages such as SPARQL and scalable endpoint architectures allow efficient queries. However, natural language is more convenient for most users. Translating natural language queries into formal language queries (e.g. SPARQL) has been a long standing artificial intelligence task. Table 1 shows an example of a natural language question with associated SPARQL query. But current systems are only successful on restricted versions of this task, e.g. using specific patterns [PHH13, TMDL17].

Since full translation-based systems are not reliable, a useful task would be the matching of related SPARQL requests (either from historical data or from the output of a translation-based system) according to their similarity to a natural language question. In this paper, we tackle the prediction of similarity between natural language questions and SPARQL requests. Word embeddings are a key component in many textual similarity systems and have been used to represent natural language questions. However, the components of SPARQL queries are either SPARQL keywords (e.g. SELECT) or Uniform Ressource Unifiers (URI) (e.g. http ://dbpedia.org/resource/Stanley_Kubrick).

There exists pre-computed URI embeddings, but learning an alignment of the embeddings latent space is needed for similarity computations, and relying on task specific manually annotated data is costly. Meta-embeddings could be used in order to solve this problem. A meta-embedding is a representation derived from a set of distinct embeddings (e.g. Word2Vec and GloVe). Yet, there exists no meta-embedding method leveraging pretrained knowledge graph embeddings and word embeddings. Such meta embedding could also allow integration of symbolic external knowledge for common sense reasoning or information retrieval.

Our contributions are as follows :

— A meta-embedding method to align word embeddings with graph entities embeddings

1

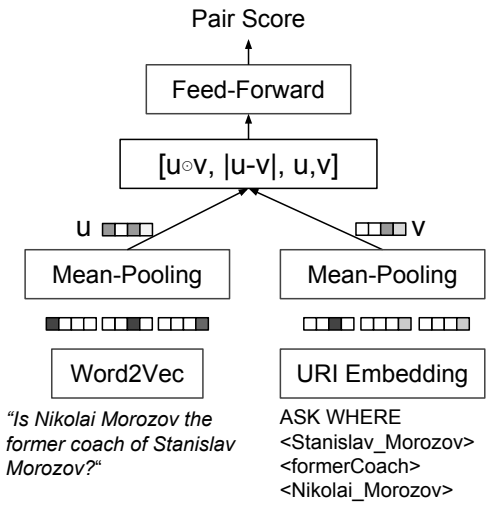


FIGURE 1 – Model architecture for similarity estimation

— Experiments on Natural Language/SPARQL queries similarity prediction
— Intrinsic evaluation of our meta-embeddings

2 Similarity estimation models

A possible use-case of our method is improvement on similarity prediction between a natural language question and a SPARQL query. In this work, we use a siamese neural network whose architecture is depicted in the figure 1 for such similarity estimation in a supervised setup. We experimented using state of the art [BSA18] models but they did not perform well, probably due to the short contexts of queries, as opposed to the evaluation datasets used in entity linking literature.

We represent the questions/queries with the average of its symbols (words/URI) embeddings, composed with a matching function (with a concatenation of hadamard product, absolute difference of input vectors) followed by a feed-forward neural network. Average-pooling is a simplistic sequence encoding method but it was shown to be competitive with more complicated architectures [SWW+18].

Representing words from natural language questions is straightforward using word embeddings. By contrast, there are several ways to represent DBPedia URI (e.g. http ://dbpedia.org/resource/Stanley_Kubrick). For instance, text can be derived from the label for the URI (e.g *Stanley Kubrick*) allowing the use of word embeddings but disregarding the knowledge from the DBPedia graph.

Pre-computed DBPedia URI embeddings [RP16] can also be used. They are embeddings computed with the SkipGram algorithm (used in Word2Vec and Node2Vec []) with DBPedia [1] graph walks instead of sentences. Such graph walks encode knowledge about entities. For example,

(Stanley_Kubrick,
writer,
A_Clockwork_Orange)

is a possible sub-path containing some useful information about Stanley Kubrick.

RDF2Vec inherits many properties from Word2Vec vectors (e.g. a cosine similarity that reflect relatedness).

In this work, we will compare the use of RDF2Vec and Word2Vec for URI representation, and propose a meta-embedding method to combine them.

3 Proposed Meta-Embedding

Word and graph embeddings encode complementary knowledge, but their latent space need to be aligned in order to perform similarity computations. Here, we propose to learn to map the latent space of RDF2Vec to the space of word embeddings.

To do so, we train a feed-forward neural network f_θ in order to predict the word embedding Word2Vec(u) representation of a given URI u from its URI embedding RDF2Vec(u). More specifically, we optimize θ in the following loss function :

$$\mathcal{L} = \sum_{u \in \mathcal{V}} \text{MSE}(\text{Word2Vec}(u), f_\theta(\text{RDF2Vec}(u))) \quad (1)$$

$\mathcal{V}$ is the set of training examples, i.e. the set of URIs where a word in a label matches a word vector. When several words are found, we use the average of their embeddings. Figure 2 illustrates this approach.

As $f_\theta(\text{RDF2Vec}(u))$ is trained to lie in the same space as Word2Vec(u), a weighted average of these representations can be also used :

$$\text{Weighted}_\alpha(u) = (1-\alpha)\text{Word2Vec}(u) + \alpha f_\theta(\text{RDF2Vec}(u)) \quad (2)$$

1. (2016-04 version)

$\text{question}_{\text{NL}}$	How many movies did Stanley Kubrick direct ?
$\text{query}_{\text{SPARQL}}$	SELECT DISTINCT COUNT(?uri) WHERE ?uri <http ://dbpedia.org/ontology/director> <http ://dbpedia.org/resource/Stanley_Kubrick>

TABLE 1 – Sample from *LC-Quad* dataset

URI Representation	Cross-Entropy	Accuracy (%)
None (Majority Class Prediction)	0.6931	90.00
Word2Vec	0.1262	97.81
g_W(RDF2vec)	0.2595	95.06
f_θ(R2Vec)	0.2610	90.57
Weighted ($\alpha = 0.075$)	**0.1189**	**97.94**

TABLE 2 – Test results of different URI embeddings models ; bold value denotes the best results *Weighted* is defined in equation 2

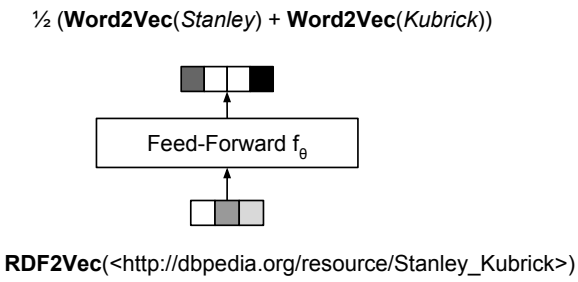


FIGURE 2 – Model architecture for embedding alignment

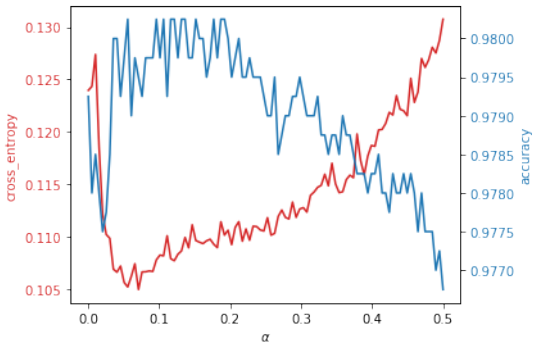


FIGURE 3 – Influence of α on matching prediction validation results

4 Experiments

4.1 Query matching evaluation

We evaluate our models on the LC-QuAD [TMDL17] dataset which is a collection of 5000 natural language questions with associated SPARQL queries. 4000 pairs are used for training and the remaining is used for evaluation. For each example, we generate 9 examples of dissimilar NL/SPARQL queries using random associations of different queries. This process is done on train data and test data separately.

To represent natural language questions, we always use word embeddings from [MGB+18] trained on CommonCrawl.

Regarding URI representations, we evaluate several embeddings :

g_W(RDF2Vec) is a linear projection of RDF2Vec embedding. The projection W is initialized randomly and learnt during the matching prediction training.

f_θ is instanciated with a two hidden layer MLP (hidden layer sizes are 200,200) with batch-normalization and ReLu activation. θ is trained on $6.0M$ URIS, using 1 epoch and using Adam optimizer [KB15] with default parameters, using the loss from equation 1. The pararameters are kept fixed in the matching prediction training.

For the matching detection training, 10% of training data is kept aside as validation set in order to determine the best number of epochs (found to be 8, also using Adam optimizer).

We performed cross validation on the parameter α. Figure 3 shows the influence on α on evaluation metrics. $\alpha = 0$ is the same as only using Word2Vec, and $\alpha = 1$ is equivalent to only using f_θ(RDF2Vec).

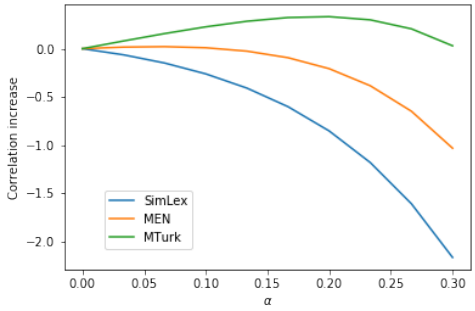


FIGURE 4 – Influence of α on word similarity prediction evaluation using the weighted combination of Word2Vec and f_θ(RDF2Vec). y axis is the pearson correlation improvement over the Word2Vec baseline.

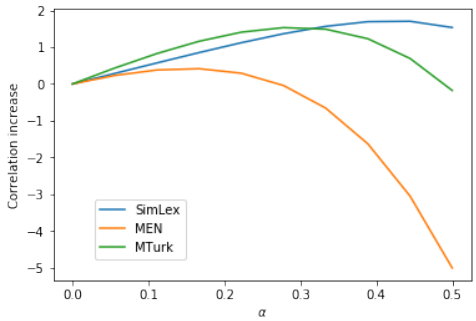


FIGURE 5 – Influence of α on word similarity prediction evaluation using the weighted combination of Word2Vec and f_θ(WordNet2Vec). y axis is the pearson correlation improvement over the Word2Vec baseline.

4.1.1 Query matching results

Table 2 shows the test results of different methods. Since accuracies are high, we also report cross entropy for a more meaningful comparison. Using the word embeddings of labels already yields high results. However, when combined with aligned graph embeddings with the *Weighted* method the results are significantly better.

4.2 Intrinsic Evaluation

We also evaluate our meta-embeddings intrinsically with a standard word similarity prediction evaluation : we use word embeddings to predict cosine similarity between word pairs, and measure the pearson correlation between cosine similarity and human judgments from similarity/relatedness prediction datasets. Sim-

	SimLex	MEN	MTurk
Word2Vec	51.8	81.7	73.3
WordNet2Vec	52.4	39.8	36.2
CONC	**53.6**	81.7	73.3
Best Weighted (RDF2Vec)	51.8	81.7	73.6
Best Weighted (WordNet2Vec)	**53.6**	**82.1**	**74.9**

TABLE 3 – Pearson correlation between cosine similarity of embeddings and human judgments for several models. We used the best values of α when reporting the score of Weighted models. CONC is a meta-ensembling baseline (concatenation of emebeddings).

Lex [HRK15] is a similarity judgement dataset (antonyms should have a low rating) while MEN [BTB14] and MTurk [RAGM11] are relatedness dataset (antonyms can have a high rating).

Once again, we use the Weighted meta-embedding model from equation 2. We report the improvement over the Word2Vec baseline according the the value of of α. Figure 5 shows the results over various datasets. We also performed the same experiment using WordNet2Vec [BAK+17] instead of RDF2Vec. WordNet2Vec is a graph embedding computed using the Wordnet graph, consisting $285k$ relations between words, such as `(furniture, is_a, piece_of_furniture)`

We used the same experimental setup but performed 2 epochs when optimizing $\mathcal{L}$. The results of best Weighted models are reported in table **??**. Our meta-embeddings are competitive with CONC while having lower dimensionality (300 vs 1150).

5 Related Work

Several models exist for meta-embedding [YS16] [MSL17]. However, they use a set of embeddings and a return a meta-embedding lying in a new latent space, except [CB18] who shows that meta-embeddings can be obtained by simply averaging or concatenating a set of input embeddings.

Retrofitting models [FDJ+15,] also improve embeddings by leveraging knowledge graphs, in a different way : they use pre-computed word embeddings and tune word representations so that they fulfill some constraints dictated by the knowledge graph.

The most similar approach to ours is[MLS13] where embeddings in different languages (e.g. french and english) are aligned using a translation matrix learn on a limited size multilingual lexicon.

The specificity of our best model is that it is additive. With proper cross validation, the weighted version of

our method can ensure better or equal results.

Another line of work deals with Alignment of knowledge from textual data and graph data. It has been explored with joint learning of embeddings from language model and knowledge graph link prediction [ABMiK18]. However, those methods are less flexible, and ca not leverage the high quality word embeddings trained on massive textual datasets without a re-training from scratch.

6 Conclusion

We proposed a simple, flexible meta-embedding method based on word embeddings and labelled graph embeddings and reported significant improvement on word representation and SPARQL queries/natural language matching. It can be applied to other graphs such as UMLS [BKF+18] for biomedical NLP or social networks graphs [LK14]. Other languages can be used as well. We expect more substantial gains on low resource languages where corpus sizes are more limited.

Acknowledgments

This work has been supported by ERA-Net CHIST-ERA LIHLITH Project funded by the Agencia Estatal de Investigación (AEI, Spain) projects PCIN-2017-118/AEI and PCIN-2017-085/AEI, the Agence Nationale pour la Recherche (ANR, France) projects ANR-17-CHR2-0001-03 and ANR-17-CHR2-0001-04, and the Swiss National Science Foundation (SNF, Switzerland) project 20CH21 174237.

Références

[ABMiK18] Mohammed Alsuhaibani, Danushka Bollegala, Takanori Maehara, and Ken ichi Kawarabayashi. Jointly learning word embeddings using a corpus and a knowledge base. In *PloS one*, 2018.

[BAK+17] Roman Bartusiak, Lukasz Augustyniak, Tomasz Kajdanowicz, Przemyslaw Kazienko, and Maciej Piasecki. Wordnet2vec : Corpora agnostic word vectorization method. *Neurocomputing*, 326-327 :141–150, 2017.

[BKF+18] Andrew L. Beam, Benjamin Kompa, Inbar Fried, Nathan P. Palmer, Xu Shi, Tianxi Cai, and Isaac S. Kohane. Clinical concept embeddings learned from massive sources of medical data. *CoRR*, abs/1804.01486, 2018.

[BSA18] Ander Barrena, Aitor Soroa, and Eneko Agirre. Learning text representations for {500k} classification tasks on named entity disambiguation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 171–180. Association for Computational Linguistics, 2018.

[BTB14] Elia Bruni, Nam-Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *J. Artif. Intell. Res.*, 49 :1–47, 2014.

[CB18] Joshua Coates and Danushka Bollegala. Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 2 (Short Papers)*, pages 194–198. Association for Computational Linguistics, 2018.

[FDJ+15] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 1606–1615. Association for Computational Linguistics, 2015.

[FEMR15] Michael Färber, Basil Ell, Carsten Menne, and Achim Rettinger. A comparative survey of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web Journal, July*, 2015.

[HRK15] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999 : Evaluating semantic models with genuine similarity estimation. *Comput. Linguist.*, 41(4) :665–695, December 2015.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

5

[LK14] Jure Leskovec and Andrej Krevl. SNAP Datasets : Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[MGB+18] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[MLS13] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013.

[MSL17] Avo Muromägi, Kairit Sirts, and Sven Laur. Linear ensembles of word embedding models. In *NODALIDA*, pages 96–104. Association for Computational Linguistics, 2017.

[PHH13] Camille Pradel, Ollivier Haemmerlé, and Nathalie Hernandez. Swip, a semantic web interface using patterns (demo) (short paper). In *International Semantic Web Conference (ISWC 2013), Sydney, Australia, 21/10/13-25/10/13*, page (electronic medium), http ://CEUR-WS.org, 2013. CEUR-WS : Workshop proceedings.

[RAGM11] Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time : Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 337–346, New York, NY, USA, 2011. ACM.

[RP16] Petar Ristoski and Heiko Paulheim. Rdf2vec : RDF graph embeddings for data mining. In *International Semantic Web Conference (1)*, volume 9981 of *Lecture Notes in Computer Science*, pages 498–514, 2016.

[SWW+18] Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. Baseline needs more love : On simple word-embedding-based models and associated pooling mechanisms. In *ACL*, 2018.

[TMDL17] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. Lcquad : A corpus for complex question answering over knowledge graphs. In *International Semantic Web Conference*, 2017.

[YS16] Wenpeng Yin and Hinrich Schütze. Learning word meta-embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 1351–1360. Association for Computational Linguistics, 2016.

Unsupervised Adversarial Image Reconstruction

Arthur Pajot*[1], Emmanuel de Bézenac*[1], et Patrick Gallinari[2]

[1]Sorbonne Universités, UMR 7606, LIP6, F-75005 Paris, France
[2]Sorbonne Universités, UMR 7606, LIP6, F-75005 Paris, France et Criteo AI Lab, Paris, France

March 25, 2019

Abstract

We address the problem of recovering an underlying signal from lossy, inaccurate observations in an unsupervised setting. Typically, we consider situations where there is little to no background knowledge on the structure of the underlying signal, no access to signal-measurement pairs, nor even unpaired signal-measurement data. The only available information is provided by the observations and the measurement process statistics. We cast the problem as finding the *maximum a posteriori* estimate of the signal given each measurement, and propose a general framework for the reconstruction problem. We use a formulation of generative adversarial networks, where the generator takes as input a corrupted observation in order to produce realistic reconstructions, and add a penalty term tying the reconstruction to the associated observation. We evaluate our reconstructions on several image datasets with different types of corruptions. The proposed approach yields better results than alternative baselines, and comparable performance with model variants trained with additional supervision.

1 Introduction

Many real world applications require acquiring information about the state of some physical system from incomplete and inaccurate measurements. For example, in infrared satellite imagery, one has to deal with the presence of clouds and a variety of other external factors perturbing the acquisition of temperature maps. This raises questions on how to recover the correct information and eliminate the contribution of external factors hindering the overall signal acquisition.

*equal contribution

In this context, signal recovery does not usually yield a unique solution, meaning that multiple signal reconstructions could trivially explain the measurements. For the above example, different missing temperature values could accurately explain the observations.

To cope with this indeterminacy, one usually relies on prior information on the structure of the true signal in order to constrain the reconstruction to plausible solutions ([21]). A common approach is to use handcrafted, analytically tractable priors ([7], [20]). This approach is limited to situations for which the underlying signal structure can be easily described, which are rarely observed in the wild.

Recent developments in generative models parameterized by neural networks ([11], [13], [10]) offer a promising statistical approach to signal recovery, for which priors on the signal are not handcrafted anymore, but learned from large amounts of data. Despite exhibiting interesting results ([4], [18], [15]), these methods all require some form of supervision, either observation measurement-signal pairs, or at least unpaired samples from observations and underlying signals. For many practical problems, obtaining these samples is too expensive and/or impractical, which makes these approaches not suitable for such situations.

We address the problem of image reconstruction in an unsupervised setting, when only corrupted observations are available, together with some prior information on the nature of the measurement process. The learning problem is formulated as finding the maximum a posteriori estimate of signals given their measurements on the training set. We derive a natural objective for our reconstruction network, composed of a linear combination of an adversarial loss for recovering realistic signals, and a reconstruction loss to tie the reconstruction to its associated observation (Section 2.2).

This model is evaluated and compared to baselines on 3 image datasets, CelebA ([17]), LSUN Bedrooms ([26]), Recipe-1M ([19]), where we experiment with different types of measurement processes corrupting the images.

Our contributions are:

- A novel, computationally efficient framework for dealing with large scale signal recovery in an unsupervised context, applicable to a wide range of situations,
- A model and a new way of training a deep learning architecture for implementing this framework,
- Extensive evaluations on a number of image datasets with different measurement processes.

2 Preliminaries

Notations. We use capital letters (*e.g.* X) for random variables, and lower-case letters (*e.g.* x) for their values. $\mathrm{p}_X(x)$ denotes the distribution (or its density in the appropriate context) of X evaluated at x.

2.1 Problem setting.

Suppose there exists a signal $X \sim \mathrm{p}_X$ we wish to acquire, but we only have access to this signal through lossy, inaccurate measurements $Y \sim \mathrm{p}_Y$. The measurement process is modeled through a stochastic operator F mapping signals X to their associated observations Y. We will refer to F as the *measurement process*, which *corrupts* the input signal. F is parameterized by a random variable $\Theta \sim \mathrm{p}_\Theta$ following an underlying distribution p_Θ we can sample from, which represents the factors of corruption. Thus, given a specific signal x, we can simulate its measurement by first sampling θ from p_Θ, and then computing $F(x;\theta)$. Additional sources of uncertainty, *e.g.* due to unknown factors, can be modeled using additive *i.i.d.* Gaussian noise $\mathcal{E} \sim \mathcal{N}(0, \sigma^2 I)$, so that the overall observation model becomes:

$$Y = F(X;\Theta) + \mathcal{E} \tag{1}$$

F is assumed to be differentiable *w.r.t.* its first argument X, and Θ and X to be independent (denoted $X \perp\!\!\!\perp \Theta$). Different instances of F will be considered (refer to Section 4.2), like random occlusions, information acquisition from a sparse subset of the signal, overly smoothing out and corrupting the original distribution with additive noise, etc. In such cases, the factors of corruption Θ might respectively represent the position of the occlusion, the coordinates of the acquired information, or simply the values of the additive noise.

2.2 Approach

Given an observation y, our objective is to find a signal $\hat{x}$ as close as possible to the associated true signal x. From a probabilistic viewpoint, it is natural to formulate the problem as finding the *maximum a posteriori* (MAP) estimate, which consists in selecting the most probable signal x^* under the posterior distribution $\mathrm{p}_{X|Y}(\cdot|y)$:

$$x^* = \arg\max_x \log \mathrm{p}_{X|Y}(x|y) \tag{2}$$

or equivalently:

$$x^* = \arg\max_x \log \mathrm{p}_{Y|X}(y|x) + \log \mathrm{p}_X(x) \tag{3}$$

where $\mathrm{p}_{Y|X}(y|x)$ is the likelihood of the signal x given observation y, and $\mathrm{p}_X(x)$ is the prior probability evaluated at x. Therefore, a good reconstruction must be likely to have generated the data, *i.e.* yield high likelihood, and look realistic, *i.e.* yield high probability under the prior.

In the general case, calculating the likelihood term $\mathrm{p}_{Y|X}(y|x)$ requires marginalizing over noise parameters Θ and this does not yield an analytic form. As for the prior $\mathrm{p}_X(x)$, it is unknown, and we have no access to samples from X since we are in an unsupervised setting: there is then no direct way to estimate p_X either. In the general case considered here, with no assumption on the form of the distributions, solving Equation (3) is up to our knowledge an open problem.

In the following sections, we will introduce an approach to deal with the likelihood term (Section 3.1), and the unknown prior term (Section 3.2) in order to provide an approximate solution to equation (3) (Section 3.3). For that, we will formulate the problem as learning a mapping $G : \mathcal{Y} \to \mathcal{X}$ that links each measurement y to its associated MAP estimate x^* on the training set. The associated objective is then:

$$G^* = \arg\max_G \mathbb{E}_{\mathrm{p}_Y} \left\{ \log \mathrm{p}_{Y|X}(y|G(y)) + \log \mathrm{p}_X(G(y)) \right\} \tag{4}$$

Which is obtained by plugging $G(y) = x$ into equation 3 and taking the expectation w.r.t. the distribution of observations p_Y.

3 Method

From Equation (4), we see that a valid reconstruction mapping G must yield high probability for the likelihood and the prior. This will guide the design of an appropriate objective during the following section, where the reconstruction mapping G will be implemented using a neural network.

3.1 Handling the Likelihood term

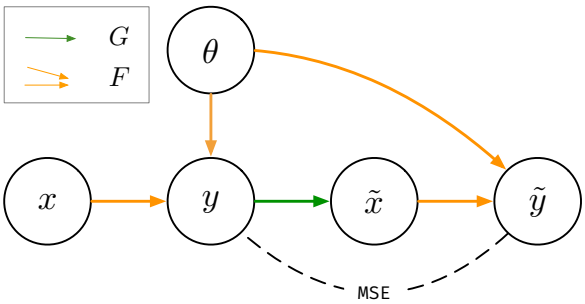


Figure 1 – The Figure illustrates the dependencies between the variables considered for handling the likelihood term when solving (4). The likelihood term in (4) can be replaced by the expectation of $\frac{1}{2\sigma^2}\|y - F(G(y);\theta)\|_2^2$ (see Equation 7). To compute this expectation, one first simulates an observation y from a signal x using $F(x;\theta)$, then generates $\tilde{x} = G(y)$ and $\tilde{y} = F(\tilde{x};\theta)$, as in the Figure. This allows us to compute the MSE term in the above expression.

In the general case, evaluating the likelihood $\mathrm{p}_{Y|X}(y|x)$ in equation (3) requires marginalizing on the unobserved noise variable Θ: $\mathrm{p}_{Y|X}(y|x) = \mathbb{E}_{\mathrm{p}_\Theta}\mathrm{p}_{Y|X,\Theta}(y|x,\theta)$, which involves computing an intractable integral. Most probabilistic model for image denoising make assumptions on the structure of the measurement operator $F(.,\Theta)$ and on the distribution of Θ in order to obtain an analytic form for the expectation ([6], [1]). Here, we consider more general measurement operators which do not necessarily lead to such a simplification and therefore proceed in a different way.

We outline below, the main steps of the method for handling the likelihood term $\mathrm{p}_{Y|X}(y|x)$ in equation (4).

1. Making use of the independence between X and Θ, the expectation term $\mathbb{E}_{\mathrm{p}_Y}\log \mathrm{p}_{Y|X}(y|G(y))$ in equation (4) can be rewritten as :

$$\mathbb{E}_{\mathrm{p}_\Theta \mathrm{p}_X \mathrm{p}_{Y|X,\Theta}} \log \mathrm{p}_{Y|X,\Theta}(y|G(y),\theta) + c_1 \qquad (5)$$

, with c_1 constant w.r.t. G.

2. The general measurement process described in equation (1) induces $\log \mathrm{p}_{Y|X,\Theta}(y|G(y),\theta)$ to yield a simple analytic expression:

$$\log p(y|G(y),\theta) = -\frac{1}{2\sigma^2}\|y - F(G(y);\theta)\|_2^2 + c_2 \qquad (6)$$

with c_2 a constant.

3. The likelihood term $\mathbb{E}_{\mathrm{p}_Y}\log \mathrm{p}_{Y|X}(y|G(y))$ can then be replaced in objective (4) by

$$-\mathbb{E}_{\mathrm{p}_\Theta \mathrm{p}_X \mathrm{p}_{Y|X,\Theta}} \frac{1}{2\sigma^2}\|y - F(G(y);\theta)\|_2^2 \qquad (7)$$

Equation (7) shows that the likelihood term can be evaluated by first sampling a measurement y conditioned on a corruption parameter θ and signal x, and then constrain G such that $\|y - F(G(y);\theta)\|_2^2$ is close to zero. Note that in this expression, the same parameter θ is used for simulating $\tilde{y}$ from $\tilde{x}$ and y from x (see Figure 1 and section 3.3 for more details). Unfortunately, this requires first sampling x from the signal distribution p_X which is unknown. In the following sections, we will see how we work around this problem.

3.2 Handling the Prior term

Maximizing w.r.t. the prior term $\mathrm{p}_X(G(y))$ in equation (4) is similar to learning a mapping G, such that the distribution induced by $G(y)$, $\mathbb{E}_{\mathrm{p}_Y}\mathrm{p}_X(G(y))$ is close to the distribution p_X. The prior p_X being unknown, the only sources of information are the lossy measurements y and the known prior p_Θ on the measurement process. In order to learn an approximation of the true prior p_X, we will use a form of generative adversarial learning, and build on an idea introduced in the AmbientGAN model by [5].

AmbientGAN aims at learning an *unconditional* generative model G of the true signal distribution p_X, when only lossy measurements y of the signal are available together with a known stochastic measurement operator F. In AmbientGAN, a generator is trained to produce uncorrupted signal samples from a latent code so that the generated signals when corrupted are indistinguishable from the observation measurements. In [5], the authors show that for some families of noise distributions p_Θ, the generator's induced distribution matches the signal's true distribution. Note that even

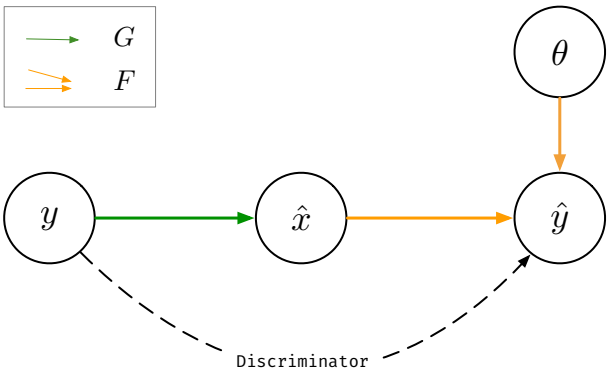


Figure 2 – The figure illustrates the dependencies of the variables used for dealing with the prior term in (4). An observation y is sampled, and then transformed by the generative network into a reconstructed signal $\hat{x} = G(y)$. One then simulates a measurement $\hat{y} := F(\hat{x}; \theta)$ from this reconstruction. We then enforce the distributions of observations p_Y and simulated measurements p_Y^G to be similar using an adversarial loss. In order to produce indistinguishable distributions, the generator G has to *remove* the corruption and recover a sample $\hat{x}$ from p_X.

if the generation process of the observations y in AmbientGAN is similar to the one considered in this paper (see Section 2.1), the objective is however different: when the aim of AmbientGAN is to learn a distribution of the underlying signal by sampling a latent space, ours is to reconstruct corrupted signals.

In order for G to produce uncorrupted signals we will use an approach inspired from AmbientGAN, as illustrated in Figure 2. Given an observation y, one wants to reconstruct a latent signal approximation $\hat{x} = G(y)$ so that a corrupted version of this signal $\hat{y} = F(\hat{x})$ will have a distribution indistinguishable from the one of the observations y. The generator G and a discriminator D are trained on observations y and generated samples $\hat{y}$. The corresponding loss is the following[1]:

$$\mathcal{L}^{\text{prior}}(G) := \max_D \; \mathbb{E}_{\substack{Y \sim \mathrm{p}_Y \\ \hat{Y} \sim \mathrm{p}_Y^G}} \left\{ \log D(y) + \log \left(1 - D(\hat{y})\right) \right\} \tag{8}$$

where p_Y^G corresponds to the distribution induced by G's corrupted outputs ($\hat{y}$ in Figure 2) , *i.e.* $\mathrm{p}_Y^G(y) := \mathbb{E}_{\mathrm{p}_\Theta \mathrm{p}_X^G} \left\{ \mathrm{p}(y|x, \theta) \right\}$ and p_X^G denotes the marginal distribution induced by G's outputs ($\hat{x}$ in Figure 2): $\mathrm{p}_X^G(x) := \mathbb{E}_{\mathrm{p}_Y} \mathrm{p}_{X|Y}^G(x|y) = \mathbb{E}_{\mathrm{p}_Y} \delta\big(x - G(y)\big)^2$. This penalty enforces the marginal p_X^G to be close to the true prior distribution p_X, and thus forces G to map its input measurements onto p_X.

[1]the min term of the adversarial loss will be introduced later, see Equation (10)

3.3 Putting everything together

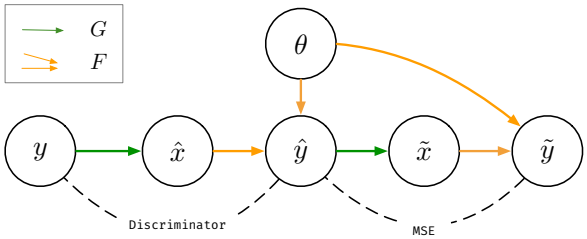


Figure 3 – General Approach. We wish to train G to recover a plausible signal from lossy measurements. As is shown in Section 2.2, this requires the reconstructions $\hat{x} := G(y)$ to have high probability under the likelihood and the prior. For simplicity, variable $\mathcal{E}$ has been omitted. *Prior :* we sample a measurement y from the data, produce a reconstruction $\hat{x}$, and sample a perturbation parameter θ. We enforce the simulated measurement $\hat{y} := F(\hat{x}; \theta)$ to be similar to measurements in the data using an adversarial penalty. Intuitively, this requires the network to *remove* the corruption. *Likelihood :* to enforce G to produce reconstructions with high likelihood, it is not possible to add a penalty to constrain the mean square error (MSE) between y and $\hat{y}$ to be small. This is because the underlying perturbation that caused y is unknown, and may be different from θ. Starting from $\hat{y}$ we generate a $\tilde{y}$ (see figure 3) using the same θ as the one used for generating $\hat{y}$. We then constrain $\|\hat{y} - \tilde{y}\|_2^2$ to be small ($\tilde{y} = F(G(\hat{y}))$).

In Section 3.1, we have shown that it is possible to maximize the average log-likelihood term in equation (4), given that we can sample from the unknown prior distribution p_X. In Section 3.2, we have shown how it is possible to enforce the generator to produce samples from p_X, without ever having access to uncorrupted samples. The idea is then to use the distribution induced by the generator's output p_X^G as a proxy for p_X to compute an approximate value of the expectation in equation (5): This gives us the following penalty term

[2]$\delta(x)$ is the Dirac delta function, which is equal to zero everywhere except in x.

:

$$\mathcal{L}^{\text{likeli}}(G) := \mathbb{E}_{\substack{\mathrm{p}_\Theta \mathrm{p}_X^G \\ \hat{Y} \sim \mathrm{p}_{Y|X,\Theta}}} \|\hat{y} - F(G(\hat{y}); \theta)\|_2^2 \quad (9)$$

The full objective is a linear combination of penalties (8) and (9):

$$\arg\min_G \ \mathcal{L}^{\text{prior}}(G) + \lambda \cdot \mathcal{L}^{\text{likeli}}(G) \quad (10)$$

As illustrated by the dependencies highlighted in Figure 2, in the process of minimizing $\mathcal{L}^{\text{prior}}$, we sample from the marginal likelihood $\mathrm{p}_Y^G(y) := \mathbb{E}_{\mathrm{p}_\Theta \mathrm{p}_X^G}\{\mathrm{p}(y|x,\theta)\}$. The expectancy in the likelihood term $\mathcal{L}^{\text{likeli}}$ is precisely computed w.r.t. this distribution. We can then use the same samples in order to minimize the full objective (10). This gives us the Algorithm 1 described below, along with the dependency structure illustrated in Figure 3.

Algorithm 1 Training Procedure.

Require: Initialize parameters of the generator G and the discriminator D.

while (G, D) not converged **do**

Sample $\{y_i\}_{1 \le i \le n}$ from data distribution p_Y

Sample $\{\theta_i\}_{1 \le i \le n}$ from P_Θ

Sample $\{\varepsilon_i\}_{1 \le i \le n}$ from $P_\mathcal{E}$

Set $\hat{y}_i$ to $F(G(y_i), \theta_i) + \varepsilon_i$ for $1 \le i \le n$

Update D by ascending:

$$\frac{1}{n}\sum_{i=1}^{n} \log D(y_i) + \log(1 - D(\hat{y}_i))$$

Update G by descending:

$$\frac{1}{n}\sum_{i=1}^{n} \lambda \cdot \|\hat{y}_i - F(G(\hat{y}_i); \theta_i)\|_2^2 + \log(1 - D(\hat{y}_i))^3$$

end while

4 Experiments

4.1 Model architectures and Datasets

Architectures. Our network architectures are inspired by the GAN architecture in [27]. We use the same discriminator, and we propose an image-to-image variant of their latent-to-image generator for the reconstruction network G.

[3] In practice we optimize $-\log D(\hat{y}_i)$ instead of $\log(1 - D(\hat{y}_i))$

Datasets. We evaluate our approach using three different image datasets :

- **CelebA.** Dataset of celebrities, containing approximately 200 000 samples. As [5], the images are center-cropped.
- **LSUN Bedrooms.** Dataset of bedrooms, containing 3 million samples.
- **Recipe-1M.** Dataset of cooked meals, containing approximately 600 000 samples.

All the images have been resized to 64×64. In order to place ourselves in the most realistic setting possible, every image has been corrupted once, *i.e.* there is never multiple occurrences of an image corrupted with different corruption parameters.

We withhold 15% of the training set for validation, selected uniformly at random for each dataset.

4.2 Corruptions

Let us present the different measurement processes F used in the experiments, also named corruptions:

Remove-Pixel. This measurement process randomly samples a fraction p of pixels uniformly and sets the associated channel values to 0. All the corresponding channel values are set to 0.

Remove-Pixel-Channel. Instead of setting to 0 a pixel for all channels as in Remove-Pixel, one samples a pixel coordinate and a channel, and sets the corresponding value to 0.

Convolve-Noise. Here $F(x; \theta) := k * x + \theta$, where $*$ is the convolution operator and k is a mean filter of size l. For each pixel, noise θ sampled from a zero-mean Gaussian of variance σ_C^2 is added to the previous result.

Patch-Band. A horizontal band of height h whose vertical position in the image is uniformly sampled from the set of possible positions. For each pixel falling inside the band, its associated value is set to 0. The resulting measurement for pixel at column i and row j can be summarized as:

$$F(x; \theta)_{i,j} := \begin{cases} 0, & \text{if } j \in \{\theta, \ldots, \theta + h\} \\ x_{i,j}, & \text{otherwise} \end{cases} \quad (11)$$

where θ is uniformly sampled from $\{1, \ldots, H - h\}$, and H is the image height. In the experiments, h is set to 20.

4.3 Baselines

4.3.1 Conditional AmbientGan.

This is our only unsupervised baseline. The context is the same as for our model: the measurement process F is assumed known, there is no access to samples from the uncorrupted signal distribution p_X, but only to their corrupted counterpart p_Y. This baseline is a combination of two recent techniques in the field of signal recovery: the aforementioned AmbientGan [5] and CS-GAN [4].

An unconditional generator G is trained using the AmbientGan framework ([5]) for each type of measurement process F, in order to produce samples from p_X (see Section 4.2). The distribution induced by the generator p_X^G is an approximation of p_X (at the optimum, both distributions match, *i.e.* $\mathrm{p}_X^G = \mathrm{p}_X$). Given a specific measurement y, the reconstruction $\hat{x}$ is the signal from G that is closest to y, as in [4]. To find $\hat{x} = G(\hat{z})$, we search for the latent code $\hat{z}$ of G, such that $\hat{z} = \arg\min_z \|y - G(z)\|_2^2 + R(z)$. $R(z)$ is a regularizing term that enforces the latent code to stay in G's input domain. This objective is optimized using stochastic gradient descent. To train G, we use the same architectures and hyper-parameters as those provided by the authors. Because this approach may be sensitive to the initial latent code, we reiterate this approach three times and select the best resulting image.

4.3.2 Unpaired Variant.

This is a variant of our model where we have access to samples of the signal distribution p_X. This means that although we have no paired samples from $\mathrm{p}_{X,Y}$, we have access to unpaired samples from p_X and p_Y. This baseline is similar to our model but instead of discriminating between a measurement from the data y and a simulated measurement $\hat{y}$, we directly discriminate between samples x from the signal distribution and the output of the reconstruction network $\hat{x}$.

4.3.3 Paired Variant.

This is a variant of our model where we have access to signal measurement pairs (y, x) from the joint distribution $\mathrm{p}_{Y,X}$. Given input measurement y, the reconstruction is obtained by regressing y to the associated signal x using a MSE loss. In order to avoid blurry samples, we add an adversarial term in the objective in order to constrain G to produce realistic samples, as in [12]. The model is trained using the same architectures as our model, and the hyperparameters have been found using cross-validation.

4.3.4 Measurement Specific Baselines.

We also compare our model to baselines that where designed to remove specific corruptions.

Deep Image Prior ([24]). Given a generator G_ϕ parametrized by randomly initialized weights ϕ and a measurement y, this method seeks to find a reconstruction from G_ϕ that is *close* to the measurement. For corruptions processes Patch-Band, Remove-Pixel and Remove-Pixel-Channel, we assume the access to the θ associated to the observations in the data (*i.e.* in this case, the mask).

Biharmonic Inpainting ([9]). By considering inpainting as a smooth surface extension domain, this baseline resolves a biharmonic equation to obtain a high order approximation of the image. This approximation is then extended to the missing part of the image. This method assumes access to the θ associated to the observations in the data (*i.e.* in this case, the mask).

Total Variation Denoising ([8]). This denoising baseline aims to minimize the total variation of an image i.e the integral of the absolute gradient of the image. Reducing the total variation of the image removes unwanted detail, such as white noise artifacts while preserving important details such as edges and corners.

5 Results

We will now present our results. First, we compare quantitatively our model with non-measurement specific baselines on CelebA. We then present qualitative results with samples from our model and these baselines.

5.1 Quantitative Results

We compare our model with baselines introduced in the previous section. We report mean square error (MSE) scores between the reconstructed $\hat{x}$ and the true signal x used to generate the input y. Table 1 shows the MSE computed on the *test* set, a randomly selected subset of CelebA comprised of 40000 images. Because the Conditional AmbientGan model is too computationally expensive, we only report the MSE on 40 randomly chosen samples of the test set.

Quantitatively, our model performs well. Except for the Conditional Ambiant GAN, all the methods are quite similar in terms of MSE. Our unsupervised model

Table 1 – : Average mean square error of neural network based models on the test set of CelebA, for different measurement processes. The first two rows are model trained with no supervision, the last two row with additional supervision.

	Remove-Pixel	Remove-Pixel-Channel	Patch-Band	Convolve-Noise
Conditional AmbientGan	0.292	0.2829	0.1421	0.0814
Our Model	**0.0414**	**0.0409**	**0.0165**	**0.0088**
Unpaired Variant	**0.037**	**0.0336**	0.034	0.0103
Paired Variant	0.0383	0.0401	**0.0147**	**0.0084**

reaches performance similar to its variants trained using additional supervision. We also note that when the aligned signal-observation pairs are not used (as in our Unpaired Variant), results are comparable – sometimes better – than when these pairs are used directly (as in our Paired Variant). This suggests that our likelihood term is sufficient to condition the reconstruction on the input signal.

5.2 Qualitative Results

However, quantitative results gives us only partial information. We now evaluate the quality of our reconstruction on three different datasets (Section 4.1). Figure 4 shows reconstructions obtained from different models on the CelebA dataset.We observe that Conditional AmbientGAN yields visually poor results, especially for the Remove-Pixel and Remove-Pixel-Channel measurement processes. We hypothesize that this is due to the large Euclidean distance between the measurements and the associated signals, and the suboptimality of the generator. Visually, the quality of our model's reconstructions are coherent with the quantitative results: they are comparable to its paired and unpaired counterparts (Section 4.3). Figures (5), (6), (7), and (8) each show reconstructions from a given measurement processe on different datasets. Our model is able to produce images with good visual quality while remaining coherent with the underlying uncorrupted images.

We can see that contrary to these methods, we are able to capture semantic information from the dataset.

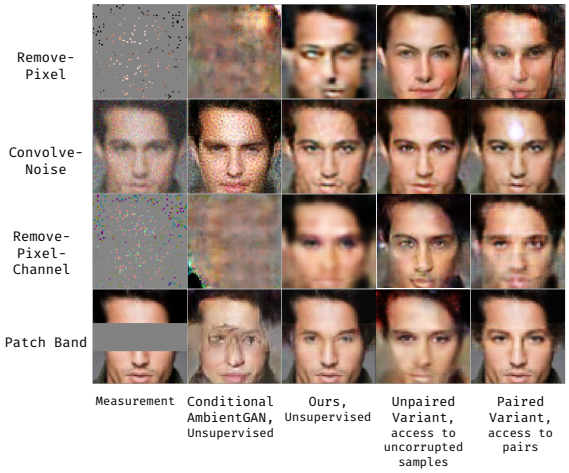


Figure 4 – Model reconstructions for different corruption processes, on CelebA. Each row corresponds to a specific corruption process, and each column to a particular model.

6 Related Work

To our knowledge, there is no other Deep Learning approach attempting to solve the unsupervised signal reconstruction problem. However, some of the ideas developed here are close to or even inspired from recent work.

In order to enforce high likelihood, we incorporate a penalty in our objective that is similar to the Cycle Consistency loss, used in several contexts ([28], [14], [2]). This constraint is used to learn from unpaired data sets. Moreover, they too use adversarial training to constrain the marginal distribution induced by the generator.

In the context of image super resolution, ([15], [22], [18]) attempt to retrieve *maximum a posteriori* estimates of the super resolution image conditioned on an input image. They too use a generative model of the signal trained in an adversarial fashion using samples from signal distribution to constrain their reconstructions. Their approach is fully supervised.

Figure 5 – On the top row, randomly sampled test set measurements from CelebA corrupted using PatchBand($h = 20$), and below, our associated reconstructions.

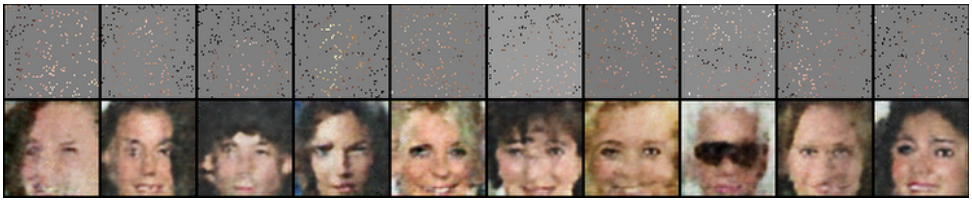


Figure 6 – On the top row, randomly sampled test set measurements from CelebA corrupted using RemovePixel($p = 0.95$), and below, our associated reconstructions.

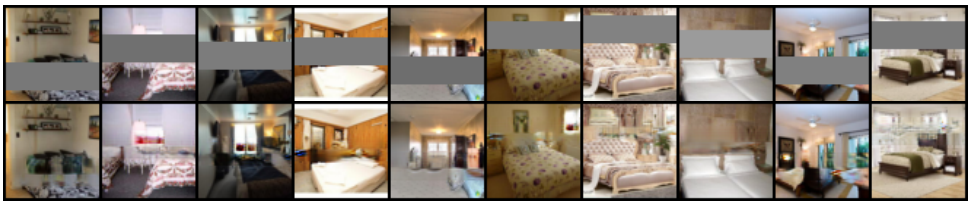


Figure 7 – On the top row, randomly sampled test set measurements from LSUN corrupted using PatchBand($h = 20$), and below, our associated reconstructions.

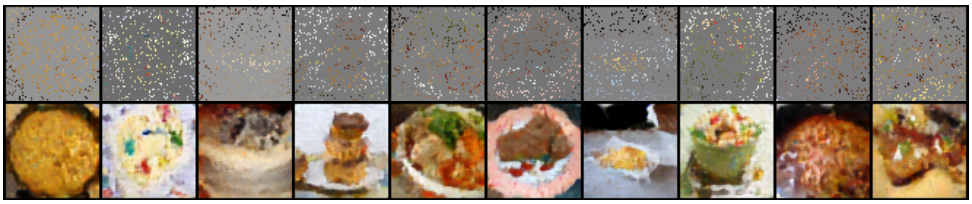


Figure 8 – On the top row, randomly sampled test set measurements from Recipe-1M corrupted using RemovePixel ($p = 0.9$), and below, our associated reconstructions.

Other works attempt to solve ill-posed inverse problems using generative models ([4], [3], [23], [25]). The general approach in all these papers consists to first train a generative model on the uncorrupted signal distribution. Then, given a measurement from which we wish to reconstruct the signal, it is *inverted* by finding the latent input code that generated the uncorrupted image, by minimizing the mean square error between the corrupted reconstruction and the measurement. This requires solving an optimization problem for each image, which takes several minutes ([24]) on GPU, and requires random restarts to avoid falling a bad local minima. Again, the setting is fully supervised.

Finally [16] propose a method for denoising images without direct supervision. They train a network to regress a corrupted image to the same image with a different corruption value. Assuming the corruption

has zero-mean, their network learns to remove the corruption by the conditional expectation. This setting implicitly assumes access to the distribution of uncorrupted images in order to generate different noisy versions of the same image, which is not our case.

7 Conclusion

We have proposed a general formulation to recover a signal from lossy measurements using a neural network, without having access to uncorrupted signal data. We have formulated the problem as finding a *maximum a posteriori* estimate of the signal given its observation, for all observations in the training set. This gives us a natural objective for our neural network, composed of a linear combination of an adversarial loss for recovering realistic signals, and a reconstruction loss to tie the reconstruction to its associated observation. Our approach yields results superior to the baselines, while staying competitive with other model variants that have access to higher forms of supervision.

For future work, we plan to apply our framework to different corruption processes, and evaluate our model's performance in real world settings, specifically for retrieving uncorrupted scientific data. Another interesting research direction would be to make our reconstruction network stochastic, in order to approximate the true posterior of the signal given the measurement, and to obtain uncertainty estimates.

Acknowledgments

This work was partially funded by ANR project LOCUST - ANR-15-CE23-0027 and by CLEAR - Center for LEArning & data Retrieval - joint lab. With Thales (`www.thalesgroup.com`).

References

[1] M. H. Alkinani and M. R. El-Sakka. Patch-based models and algorithms for image denoising: a comparative review between patch-based images denoising methods for additive noise reduction. *EURASIP Journal on Image and Video Processing*, 2017(1):58, Aug 2017.

[2] A. Almahairi, S. Rajeswar, A. Sordoni, P. Bachman, and A. Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. *arXiv preprint arXiv:1802.10151*, 2018.

[3] M. Asim, F. Shamshad, and A. Ahmed. Solving bilinear inverse problems using deep generative priors. *CoRR*, abs/1802.04073, 2018.

[4] A. Bora, A. Jalal, E. Price, and A. G. Dimakis. Compressed Sensing using Generative Models. *arXiv:1703.03208 [cs, math, stat]*, Mar. 2017. arXiv: 1703.03208.

[5] A. Bora, E. Price, and A. G. Dimakis. AmbientGAN: Generative models from lossy measurements. In *International Conference on Learning Representations*, 2018.

[6] A. K. Boyat and B. K. Joshi. A review paper: Noise models in digital image processing. *CoRR*, abs/1505.03489, 2015.

[7] E. J. Candès, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2005.

[8] A. Chambolle. An Algorithm for Total Variation Minimization and Applications. *Journal of Mathematical Imaging and Vision*, 20(1):89–97, jan 2004.

[9] S. Damelin and N. Hoang. On surface completion and image inpainting by biharmonic functions: Numerical aspects. *International Journal of Mathematics and Mathematical Sciences*, 2018, 2018.

[10] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. *CoRR*, abs/1605.08803, 2016.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[12] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[13] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[14] G. Lample, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. *CoRR*, abs/1711.00043, 2017.

[15] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[16] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila. Noise2noise: Learning Image Restoration without Clean Data. *arXiv:1803.04189 [cs, stat]*, Mar. 2018. arXiv: 1803.04189.

[17] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[18] M. Mardani, E. Gong, J. Y. Cheng, S. Vasanawala, G. Zaharchuk, M. T. Alley, N. Thakur, S. Han, W. J. Dally, J. M. Pauly, and L. Xing. Deep generative adversarial networks for compressed sensing automates MRI. *CoRR*, abs/1706.00051, 2017.

[19] J. Marin, A. Biswas, F. Ofli, N. Hynes, A. Salvador, Y. Aytar, I. Weber, and A. Torralba. Recipe1m: A dataset for learning cross-modal embeddings for cooking recipes and food images. *arXiv preprint arXiv:1810.06553*, 2018.

[20] J. F. Mota, N. Deligiannis, and M. R. Rodrigues. Compressed sensing with prior information: Strategies, geometry, and bounds. *IEEE Transactions on Information Theory*, 63(7):4472–4496, 2017.

[21] A. M. Stuart. Inverse problems: A bayesian perspective. *Acta Numerica*, 19:451–559, 2010.

[22] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised MAP Inference for Image Super-resolution. *arXiv:1610.04490 [cs, stat]*, Oct. 2016. arXiv: 1610.04490.

[23] S. Tripathi, Z. C. Lipton, and T. Q. Nguyen. Correction by projection: Denoising images with generative adversarial networks. *CoRR*, abs/1803.04477, 2018.

[24] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Deep image prior. *CoRR*, abs/1711.10925, 2017.

[25] D. Van Veen, A. Jalal, E. Price, S. Vishwanath, and A. G. Dimakis. Compressed Sensing with Deep Image Prior and Learned Regularization. *arXiv:1806.06438 [cs, math, stat]*, June 2018. arXiv: 1806.06438.

[26] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015.

[27] H. Zhang, I. J. Goodfellow, D. N. Metaxas, and A. Odena. Self-attention generative adversarial networks. *CoRR*, abs/1805.08318, 2018.

[28] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv:1703.10593 [cs]*, Mar. 2017. arXiv: 1703.10593.

On Fair Cost Sharing Games in Machine Learning

Ievgen Redko[1] and Charlotte Laclau[1]

[1]Univ Lyon, UJM-Saint-Etienne, CNRS, Institut d Optique Graduate School, Laboratoire Hubert Curien UMR 5516, F-42023, Saint-Etienne, France

Abstract

Machine learning and game theory are known to exhibit a very strong link as they mutually provide each other with solutions and models allowing to study and analyze the optimal behaviour of a set of agents. In this paper, we take a closer look at a special class of games, known as *fair cost sharing games*, from a machine learning perspective. We show that this particular kind of games, where agents can choose between selfish behaviour and cooperation with shared costs, has a natural link to several machine learning scenarios including collaborative learning with homogeneous and heterogeneous sources of data. We further demonstrate how the game-theoretical results bounding the ratio between the best Nash equilibrium (or its approximate counterpart) and the optimal solution of a given game can be used to provide the upper bound of the gain achievable by the collaborative learning expressed as the expected risk and the sample complexity for homogeneous and heterogeneous cases, respectively. We believe that the established link can spur many possible future implications for other learning scenarios as well, with privacy-aware learning being among the most noticeable examples.

1 Introduction

In recent years, machine learning community witnessed an increasing interest among the researchers towards the game theory, a sub-field of mathematics that studies the problem of decision-making in the presence of various types of constraints. This is no surprise as game theory proposes tools and a large variety of theoretical results allowing to determine optimal strategies of a set of agents that may have conflicting or collaborative objectives: a situation commonly encountered in many machine learning multi-objective optimization problems. This kind of problems, for instance, can be faced by game playing machine learning algorithms that seek to find an optimal trade-off between successfully passing the game level, earning the highest number of bonuses and doing all this in the fastest possible way. To this end, game theory has become a topic of ongoing interest in machine learning field that has already found its application in contributions related to numerous learning scenarios such as reinforcement learning [PKMK00, HW03, CB98, PL05], supervised learning [FS99, SSS07a, SSS07b, SZ16], and adversarial classification [LC09, BS11, DLM17] to name a few.

In this paper, we take a closer look at *fair cost sharing games*, a special case of games that studies the optimal strategy of a set of agents when they can choose between a cooperative behaviour with a cost equally shared among them and a non-cooperative (also called "selfish") behaviour. As an example of this game, one can consider a set of colleagues that face a choice between driving their cars to work and individually paying for it or taking the bus together and sharing the cost of the trip. We show that this particular game can be naturally related to collaborative learning, a problem often encountered in machine learning that consists in finding the best hypothesis for a set of data samples drawn from (possibly) different probability distributions and achieving a nearly optimal individual performance with respect to some task. In this setting, we associate each agent with a data sample that it can share with other agents in order to learn in a collaborative way on a larger concatenated sample or stick to what it has and learn on the sample available to it. We analyze this problem in several settings where the agents' sample may or may not be drawn from the same probability distribution and where each agent may have a weight that corresponds to to its potential benefit from collaboration and its contribution to it. For both cases considered, we propose a theoretical result that bounds the ratio between the overall cost of non-collaborative learning with respect to the collaborative one where the cost can be defined based

on empirical risk achieved by the optimal hypothesis or on the sample complexity of the considered learning approach. To the best of our knowledge, this is the first contribution that establishes a connection between this class of games and collaborative learning and shows its usefulness in providing new theoretical guarantees for the latter.

The rest of this paper is organized as follows. We first introduce the related works that exist in the literature providing the theoretical analysis of the considered learning settings both in traditional and game-theoretical contexts. Then, we present necessary background definitions, on which we rely in the following sections, that introduce both basic and weighted versions of the fair cost sharing games and theoretical results established for them. We further proceed by first formally describing the considered setup and then by presenting our main contributions based on it. The last section of this paper is devoted to conclusions and to the description of several future perspectives of this work.

2 Related Works

Despite a considerable amount of work situated at the intersection of game theory and machine learning mentioned in the previous section, few of them are related to the main contributions of this paper in terms of the quantities of interest that they analyze and the class of games that they consider. We present the related work structured with respect to these criteria below.

Mechanism design via machine learning Arguably, one of the first papers that analyzed a certain class of games using the concepts from statistical learning theory and sample complexity in particular was presented in [BBHM05][1]. Their contribution considered a revenue-maximizing game where the goal is to find an optimal pricing function for a set of auction bidders, and consisted in showing that the optimal solution of this problem can be characterized using the techniques from statistical learning theory. We, however, consider a different class of games and, more importantly, undertake the opposite direction that consists in providing new results for collaborative machine learning problems using game-theoretical concepts. As this research direction is in general quite unrelated to ours, we refer the interested reader to [LCQ15, Related work] for a more complete up-to-date survey on the subject.

[1]This paper is a follow-up work of [BKRW03, BH05] where the same problem was considered in the context of online learning but without relying on the sample complexity results.

Statistical cost sharing Several recent papers [BPZ15, BSV17] considered the class of cost sharing games that present the main subject of investigation of this paper. The main goal of [BPZ15] was to define an algorithmic approach with strong theoretical guarantees that allows to calculate the cost-sharing function and define the optimal costs of agents based on it. The authors of [BSV17] improved the analysis provided in [BPZ15] and also addressed the estimation of the Shapley value [Sha53], a unique vector of cost shares that satisfies a set of natural axioms [HSE95]. These papers are similar to ours as they relate the probably approximately correct (PAC) analysis [Val84] to cost sharing games. However, our work differs from these latter in two principal ways: (1) while cited papers aim to find an algorithmically optimal way to calculate the costs of collaboration for each agent satisfying several natural axioms, we consider a special case of cost sharing games with a fair division scheme; (2) similar with mechanism design contributions mentioned above, our paper aims at applying results from game theory to provide new insights for PAC analysis of collaborative learning, while [BPZ15, BSV17] tailor traditional PAC analysis to study cost sharing games.

Strategyproof classification Strategy-proof classification problem studied in [DFP10, MPR12] deals with a collaborative learning setup where a group of agents have a choice between reporting their true labels or falsifying them in order to achieve a better individual classifier. For this problem, the above-mentioned papers showed that, under some assumptions, the popular empirical risk minimizing (ERM) mechanism is truthful and optimal, *i.e.*, it encourages all agents to report their true labels and provides an approximately optimal solution for all of them. Our work is close to this line of research as it also considers empirical risk as a cost of collaboration for each agent and studies the general collaborative learning scenario in the PAC setting. Despite this similarity, the purpose of our work is different as we aim to study the gain possibly achievable by a collaborative learning algorithm: a question that was not addressed by either of these works.

Collaborative learning Finally, we briefly cover the contributions that establish theoretical guarantees for collaborative learning setting considered in this pa-

per, where a set of agents aim at learning an accurate model simultaneously. The notion of collaborative learning is very vast and covers such areas as multi-task learning [Bax97, Car97, KI12], multi-source domain adaptation [BDBC+10, MMR09a, MMR09b] and distributed learning [BBFM12, WKS16]. To this end, we note that our work is similar to that presented in [BHPQ17] where the authors seek to establish the approximate value of the ratio between the sample complexity of non-collaborative learning and collaborative learning settings: a quantity denoted by them as overhead. The authors further propose an algorithm and a PAC analysis showing that its overhead is logarithmic. Our contribution completes this analysis with an upper-bound of the overhead and provides this result in a different, and arguably simpler, way. We elaborate more on the link between the two results in the section where the main contributions of our paper are presented.

3 Preliminary Knowledge

In this section, we present the preliminary knowledge related to the game-theoretic concepts that we use later in this paper. We start with a general description of a game and proceed by introducing a fair cost sharing game and some results obtained for it.

3.1 General Definitions

Given N, a set of K agents, $\mathcal{S}_i$, a finite action space of agent $i \in N$ and c_i, a cost function of agent i, a game is defined as a tuple

$$\mathcal{G} = \langle N, (\mathcal{S}_i), (c_i) \rangle .$$

We define the joint action space of the agents as $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_K$ and let the cost function c_i associated to an agent i be a mapping of a joint action $s \in \mathcal{S}$ to a real non-negative number, *i.e.*, $c_i : \mathcal{S} \to \mathbb{R}^+$. In this work, we assume that the social cost $c : \mathcal{S} \to \mathbb{R}^+$ is defined as the overall sum of agent's costs $\sum_{i=1}^{K} c_i$. The optimal social cost for this scenario is given by the strategy minimizing the cost c:

$$\mathbf{OPT}(\mathcal{G}) = \min_{s \in \mathcal{S}} c(s).$$

In this game, we say that a joint action $s \in \mathcal{S}$ (also called strategy) is a *pure Nash Equilibrium* [Nas50, Nas51] if no agent $i \in N$ can benefit from unilaterally deviating to another action. Denoting by $N(\mathcal{G})$ the set of Nash equilibria of the game $\mathcal{G}$, we further define two key quantities related to games as follows.

Definition 1. *The Price of Anarchy (PoA) is the ratio of the worst Nash equilibrium to the social optimum. It measures how the efficiency of a system deteriorates due to a selfish behavior of the agents of the game,* ***i.e.****,*

$$PoA(\mathcal{G}) = \max_{s \in N(\mathcal{G})} c(s)/\mathbf{OPT}(\mathcal{G}).$$

Definition 2. *The Price of Stability (PoS) is the ratio of the best Nash equilibrium to the social optimum, i.e.,*

$$PoS(\mathcal{G}) = \min_{s \in N(\mathcal{G})} c(s)/\mathbf{OPT}(\mathcal{G}).$$

The motivation behind introducing PoS in addition to PoA stems from the fact that this latter can be very large, making it uninformative in practice. As we show it below, this is the case for a particular class of games studied in this paper.

We also note that some classes of games do not admit a pure Nash equilibrium [ADTW03] so that an approximate Nash equilibrium should be considered. This latter can be defined as a strategy for which no agent can decrease its cost by more than an α multiplicative factor from unilaterally deviating to another action. In this case, the quantities PoS and PoA are defined with respect to the α-approximate Nash equilibrium in the same manner.

3.2 Fair Cost Sharing Game

In this paper, we focus on a specific class of games, referred to as *fair cost sharing games*. Such a game take place in a graph $G = (V, E)$ with a set of K agents, where each edge $e \in E$ carries a non-negative cost γ_e, and each agent i has source node $s_i \in V$ and destination node $t_i \in V$ that it tries to connect. We denote by $\mathcal{S}_i$ the set of paths taken by agent i in order to connect s_i to t_i. Outcomes of the game correspond to path vectors $s = (P_1, \ldots, P_K)$, with each agents choosing a single path $P_i \in \mathcal{S}_i$. We further denote by x_e the number of agents whose strategy contains edge e.

One can think of γ_e as the fixed cost of building the edge e, and this cost is independent of the number of agents that use the edge. Therefore, if more than one agent use an edge e in their chosen paths, *i.e.*, $x_e > 1$, then they share the edge's fixed cost γ_e. In a *fair cost sharing game*, we assume that the cost is split equally among the agents meaning that the cost to any agent i is

$$c_i(s) = \sum_{e \in P_i} \frac{\gamma_e}{x_e}.$$

Then, the objective is to minimize the total cost of the formed network defined as

$$c(P_1, \ldots, P_K) = \sum_{e \in \bigcup_i P_i} \gamma_e.$$

As an illustrative example, one can consider a special instance of a fair cost sharing game, referred to as opting out, presented in Figure 1. Here, the K agents have distinct sources $s_1, \ldots, s_K$, but a common destination t. In order to reach t, they have two options: (1) meeting at a rendezvous point v and continuing together to t, resulting in a joint cost of $1 + \varepsilon$, for some small $\varepsilon > 0$ or (2) taking the direct $s_i - t_i$ path individually. In this case, each agent i incurs a cost of $\frac{1}{i}$ for its opt-out strategy leading to a unique Nash equilibrium with cost $\mathcal{H}_K = \sum_{i=1}^{K} \frac{1}{i}$. However, one can clearly observe that the optimal solution in this game would be for all players to travel through the rendezvous point for an overall total cost of $1 + \varepsilon$ incurring an individual cost of $\frac{1+\epsilon}{K}$ for each agent. Furthermore, in the worst case, Nash equilibria of this game can be very expensive, so that the PoA becomes as large as K. To see this, we can consider a graph with common source and destination nodes for all agents and two parallel edges of cost 1 and K between them. In this case, the worst equilibrium corresponds to all players choosing the more expensive edge and paying K times the cost of the optimal solution. In its turn, PoS can be bounded due to the following theorem.

Theorem 1 ([ADK+04]). *The PoS for pure Nash equilibria in fair cost sharing games is at most $\mathcal{H}_K = \Theta(\log K)$, where $\mathcal{H}_K = 1 + \frac{1}{2} + \ldots + \frac{1}{K}$ and this bound is tight.*

This theorem provides us with a trade-off that can exist between the cost of selfish behaviour related to a pure Nash equilibrium for a set of agents and that of an optimal social cost. As shown in [ADK+04], this bound is not vacuous as one may always find an example of a game where the ratio between the two is exactly $\Theta(\log K)$.

Note that the fair cost sharing game presented above admits that every agent pays the same cost for using the shared edge, even though in many real-world applications one may expect the cost to be shared among agents depending on their weights $\{w_i\}_{i=1}^{K}$. This weight, for instance, can be related to the contribution of a given agent in the collaboration. This scenario leads to a *weighted fair cost sharing game*, where the cost of each agent i is proportional to its

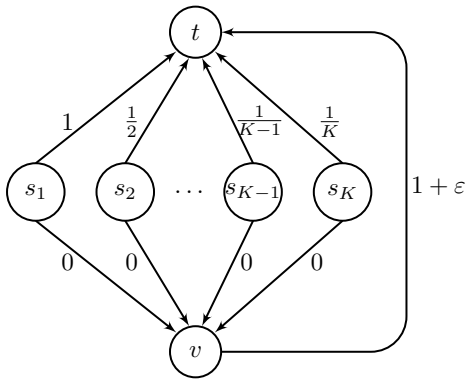


Figure 1: Example of cost-sharing network known as opting out. Here the cost of Nash equilibrium is $\mathcal{H}_K = \sum_{i=1}^{K} \frac{1}{i}$ while the optimal cost is $1 + \epsilon$.

weight w_i and can be calculated as follows:

$$c_i(s) = \sum_{e \in P_i} \gamma_e \frac{w_i}{W_e}$$

with W_e denoting the total weight of the players that select a path containing e.

For this particular case, the pure Nash equilibrium exists only in games with 2-players [ADK+04] and thus α-approximate Nash equilibria are usually considered. Before presenting the theorem that bounds the PoS of weighted fair cost sharing games, we first assume that for all i, $w_i \geq 1$ and denote by $w_{\max} = \max_{i \in [1,\ldots,K]} w_i$ the maximum weight across all agents. We further let $W = \sum_{i=1}^{K} w_i$ be the overall sum of weights of all agents. The desired result can be now stated as follows.

Theorem 2 ([CR09]). *For $\alpha = \Omega(\log w_{max})$, every weighted fair cost sharing game admits a $\mathcal{O}(\alpha)$-approximate Nash equilibrium for which the PoS is at most $\mathcal{O}\left(\frac{\log(W)}{\alpha}\right)$.*

With these two theorems, we are now ready to present our main contributions.

4 Main Contributions

In this section, we present our main contribution that consists in showing how different learning paradigms can be seen as instances of fair cost sharing games. Our goal here is two-fold and consists in showing that: (1) representing different learning scenarios as instances of fair cost sharing games establishes a connection between the statistical learning theory and the game theory; (2) this connection can be used to inherit some

important guarantees established in the rich literature on game theory. With this in mind, we now proceed to a formal description of the considered setup.

4.1 Problem Setup

Let us consider a set of K agents $\{a_i\}_{i=1}^K$, where each agent has access to a learning sample $S^i = \{(\mathbf{x}_j^{(i)}, y_j^{(i)})\}_{j=1}^{m_i}$ of size m_i, for all $i \in [1, \dots, K]$. For each i, we assume that S^i is drawn i.i.d. from a probability distribution $\mathcal{D}^i$ defined over a product space $\mathbf{X} \times Y$, where $\mathbf{X} \subseteq \mathbb{R}^d$ and Y is an output space that can be equal to $\{0, 1\}$ in case of binary classification. In practice, S^i can be given by a collection of images, while classes $\{0, 1\}$ may define the presence or absence of a certain object on an image. For a convex loss function $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ and a sample S^i, we define the true and empirical risks for each $i \in [1, \dots, K]$ as follows:

$$\begin{aligned}
\mathbb{R}_{\mathcal{D}^i}(h(\mathbf{x}), y) &= \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim\mathcal{D}^i} [\ell(h(\mathbf{x}), y], \\
\mathbb{R}_{\hat{\mathcal{D}}^i}(h(\mathbf{x}), y) &= \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim\hat{\mathcal{D}}^i} [\ell(h(\mathbf{x}), y)] \\
&= \frac{1}{m_i} \sum_{j=1}^{m_i} \ell(h(\mathbf{x}_j^{(i)}), y_j^{(i)}),
\end{aligned}$$

where $\hat{\mathcal{D}}^i = \frac{1}{m_i} \sum_{i=1}^{m_i} \delta_{\mathbf{x}_j^{(i)}}$ is an empirical distribution associated with $\mathcal{D}^i$ and $h \in \mathcal{H}$ is a hypothesis from some hypothesis space $\mathcal{H}$ such that $h : \mathbf{X} \to Y$. As an example, one may consider $\mathcal{H}$ as a space of linear functions so that h would be a hyperplane that separates the two classes. We now define the risk-minimizing hypothesis as follows: let us denote by $h^*_{S^i} = \operatorname{argmin}_{h\in\mathcal{H}} \mathbb{R}_{\hat{\mathcal{D}}^i}(h(\mathbf{x}), y)$ and $h^*_{\mathcal{D}^i} = \operatorname{argmin}_{h\in\mathcal{H}} \mathbb{R}_{\mathcal{D}^i}(h(\mathbf{x}), y)$ for all $i \in [1, \dots, K]$ the empirical and true risk-minimizing hypotheses, respectively. We further denote by $\hat{\mathbb{R}}^*_{S^i}(\mathcal{H}) = \mathbb{R}_{\hat{\mathcal{D}}^i}(h^*_{S^i}(\mathbf{x}), y)$ and $\mathbb{R}^*_{\mathcal{D}^i}(\mathcal{H}) = \mathbb{R}_{\mathcal{D}^i}(h^*_{\mathcal{D}^i}(\mathbf{x}), y)$ the ideal empirical and true risks, respectively.

4.2 Collaborative Learning with Homogeneous Sources

In order to present our first result, we start by considering a traditional collaborative learning setting where agents have access to learning samples S^i drawn from the same underlying probability distribution $\mathcal{D} = \mathcal{D}^i, \forall i \in [1, \dots, K]$. For instance, this scenario can occur in practice when several hospitals build predictive models based on their collected data consisting of annotated MRI scans: if the scanners used to produce images are the same, we can suppose that the statistical distribution of images related to the same organ is also highly similar. Bearing in mind the high cost of manual labeling of MRI scans required to increase the sample size so that a low-error hypothesis can be learned, the hospitals may think of joining their forces and pooling their labeled samples together. In this case, the goal of our analysis would be to derive a bound on the ratio between the overall performance achieved by individual agents that learn on a limited sample available and the performance of a hypothesis obtained using a larger sample $S = \bigcup_{i=1}^K S^i$.

In order to make the considered problem more realistic, we attribute weights to each agent reflecting the number of labeled instances that it provides to the collaborative learning algorithm. Intuitively, the cost of collaboration for agents that have large data samples should be smaller as they benefit less from collaboration due to their capacity of being able to learn a good classifier on their own. To this end, we define the weights w_i of agents for all $i \in [1, \dots, K]$ as a ratio $\frac{m_{\max}}{m_i}$, where $m_{\max} = \max_{i\in[1,\dots,K]} m_i$. This definition ensures that the weight of the agent having access to the largest sample is equal to 1, while for all the others it is greater or equal than 1. We can now state the following theorem.

Theorem 3. *Assume that for all $i \in [1, \dots, K]$, $\mathcal{D} = \mathcal{D}^i$. Let $h^*_S = \operatorname{argmin}_{h\in\mathcal{H}} \mathbb{R}_{\hat{\mathcal{D}}}(h(\mathbf{x}), y)$, where $\hat{\mathcal{D}}$ is an empirical distribution associated with the sample $S = \bigcup_{i=1}^K S^i$, such that $|S| = \sum_{i=1}^K m_i = m$ for a hypothesis space $\mathcal{H}$. Let $\hat{\mathbb{R}}^*_{\mathcal{D}^i_S}(\mathcal{H}) = \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim\hat{\mathcal{D}}^i} [\ell(h^*_S(\mathbf{x}), y]$ and assume further that $\sum_{i=1}^K \hat{\mathbb{R}}^*_{S^i}(\mathcal{H}) \geq \sum_{i=1}^K \hat{\mathbb{R}}^*_{\mathcal{D}^i_S}(\mathcal{H}) \geq \frac{1}{\alpha} \max_{i\in[1,\dots,K]} \hat{\mathbb{R}}^*_{S^i}(\mathcal{H})$ for some $\alpha \geq 0$ with $\hat{\mathbb{R}}^*_{S^i}(\mathcal{H}) > 0$ for all $i \in [1, \dots, K]$. Then, the following holds:*

$$\frac{\sum_{i=1}^K \hat{\mathbb{R}}^*_{S^i}(\mathcal{H})}{\sum_{i=1}^K \hat{\mathbb{R}}^*_{\mathcal{D}^i_S}(\mathcal{H})} \leq \mathcal{O}\left(\frac{\log\left(\sum_{i=1}^K \frac{m_{max}}{m_i}\right)}{\alpha}\right).$$

Proof. The main idea of our proof is to show that this particular learning setting can be represented as an instance of a weighted fair cost sharing game. Once this is done, we can simply apply Theorem 2 to obtain the desired result.

To this end, we start by considering the construction represented in Figure 2 and proceed by defining

the nodes and edges with their associated costs as follows. First, we let the nodes a_i correspond to source nodes of agents $\{a_i\}_{i=1}^K$ with their respective learning samples S^i. The node $\mathbf{L}$ corresponds to the destination node where a risk-minimizing classifier is learned, while the node $\mathbf{P}$ corresponds to pooling the data from the incoming edges. In this game, each agent a_i has a choice between learning a classifier using its own available sample S^i by taking the edge $a_i - \mathbf{L}$ or pooling it with other agents by choosing the path $a_i - \mathbf{P} - \mathbf{L}$. This latter choice stands for learning a classifier that minimizes the overall error on the union of their samples with an individual cost of $c^* \frac{w_i}{\sum_{i=1}^K w_i}$. In this case, we can define the costs of edges as follows: we assume that the cost of taking the edge from node a_i to $\mathbf{L}$ has a cost of the optimal risk achievable by minimizing it over the observable sample S^i. Thus, we write for all $i \in [1, \ldots, K]$, $c_i = \hat{\mathbb{R}}^*_{S^i}(\mathcal{H})$.

As we consider a scenario where all the data distributions $\mathcal{D}^i$ are the same, it is reasonable to further assume that the price of pooling the data is equal to 0 for each agent as it does not require reducing the discrepancy between the different agents' distributions. In this case, the price of taking the edge between each node a_i and $\mathbf{P}$ is set to 0.

The price of learning in a collaborative way can be characterized by the sum of optimal risks achieved for each agent with respect to the hypothesis minimizing the risk on sample S. Consequently, we define it by letting $c^* = \sum_{i=1}^K \hat{\mathbb{R}}^*_{\mathcal{D}^i_S}(\mathcal{H})$. As in the classical fair cost sharing game, each agent has a preference for choosing a "selfish" non-collaborative strategy that consists in learning using its own sample when the inequality $\sum_{i=1}^K \hat{\mathbb{R}}^*_{\mathcal{D}^i_S}(\mathcal{H}) \geq \frac{1}{\alpha} \max_{i \in [1,\ldots,K]} \hat{\mathbb{R}}^*_{S^i}(\mathcal{H})$ holds. This latter condition corresponds to a α-approximate Nash equilibrium of this game that has a cost of $\sum_{i=1}^K \hat{\mathbb{R}}^*_{\mathcal{D}^i}(\mathcal{H})$. From the assumption of the theorem, the optimal solution, however, is to learn on a bigger sample by following the path $a_i - \mathbf{P} - \mathbf{L}$ with a cost of $\sum_{i=1}^K \hat{\mathbb{R}}^*_{\mathcal{D}^i_S}(\mathcal{H})$ that is shared by all agents. Applying Theorem 2, we can bound the ratio between the overall cost achieved at α-approximate Nash equilibrium and the optimal solution as follows:

$$\frac{\sum_{i=1}^K \hat{\mathbb{R}}^*_{\mathcal{D}^i}(\mathcal{H})}{\sum_{i=1}^K \hat{\mathbb{R}}^*_{\mathcal{D}^i_S}(\mathcal{H})} \leq \mathcal{O}\left(\frac{\log\left(\sum_{i=1}^K \frac{m_{\max}}{m_i}\right)}{\alpha}\right).$$

□

The result established in this theorem states that for a considered learning scenario where additional data coming from K different sources can help to learn a better classifier, the gap between the sum of individual performances of all agents and that achieved by the collaborative learning approach increases when there exist important differences between the sample sizes of different agents. This implication is not trivial but rather intuitive as we may expect to obtain an improved performance at least for those agents who possess little data due to the data provided by other agents with bigger data samples. Note also that the statement of the theorem can be further simplified if $\alpha = 1$, *i.e.* α-approximate Nash equilibrium coincides with the pure one, but this latter exists only when $K = 2$.

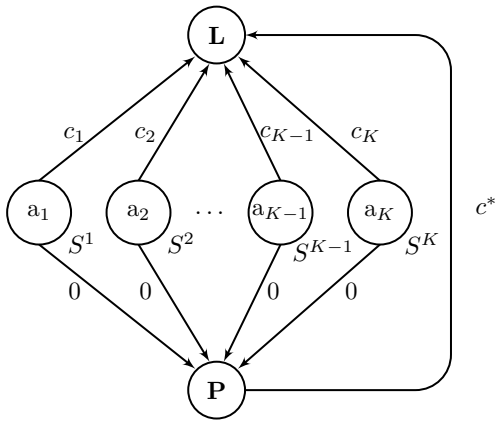


Figure 2: (**Theorem 3**) Collaborative learning as a fair cost sharing game with multiple agents corresponding to different data sources generated from the same probability distributions. Here, for all $i \in [1, \ldots, K]$, $c_i = \hat{\mathbb{R}}^*_{S^i}(\mathcal{H})$ while $c^* = \sum_{i=1}^K \hat{\mathbb{R}}^*_{\mathcal{D}^i_S}(\mathcal{H})$; (**Theorem 4**) Collaborative learning with data sources generated from different probability distribution. Here, for all $i \in [1, \ldots, K]$, $c_i = m^i_{\epsilon,\delta} = m_{\epsilon,\delta}$ while $c^* = m^*_{\epsilon,\delta}$.

Before proceeding to the analysis of a more general collaborative learning scenario, we first briefly comment on the main assumption of the theorem given by the inequality

$$\sum_{i=1}^K \hat{\mathbb{R}}^*_{S^i}(\mathcal{H}) \geq \sum_{i=1}^K \hat{\mathbb{R}}^*_{\mathcal{D}^i_S}(\mathcal{H}) \geq \frac{1}{\alpha} \max_{i \in [1,\ldots,K]} \hat{\mathbb{R}}^*_{S^i}(\mathcal{H}). \quad (1)$$

First, we note that the left-hand side of (1) implies that a hypothesis learned on a bigger sample performs better on each individual sample of all agents a_i. Even though it can be violated in some real-world scenarios, this assumption is rather intuitive in the framework of supervised learning where the generalization capacity

tends to increase with the increasing sample size for a fixed hypothesis space. As for the right-hand side, the inequality $\sum_{i=1}^{K} \hat{\mathbb{R}}^*_{\mathcal{D}^i_S}(\mathcal{H}) \geq \frac{1}{\alpha} \max_{i\in[1,\ldots,K]} \hat{\mathbb{R}}^*_{S^i}(\mathcal{H})$ means that the overall risk related to collaborative learning is higher than the worst performance over all agents multiplied by a factor of $\frac{1}{\alpha}$. It further implies that for all $i \in [1,\ldots,K]$, $c_i \leq \alpha c^*$ meaning that the dominant strategy for all agents corresponding to the α-approximate Nash equilibrium is to learn using their own sample. This assumption restricts agents from deviating from their non-collaborative strategy but, as it was seen from the example presented in Figure 1, does not imply that the individual costs of all agents c_i are lower than the cost of a sharing edge c^*/K. In order to see this, one can consider an arbitrary vector $[c_1,\ldots,c_K]$ with $\forall i \in [1,\ldots,K], c_i > 0$ and denote by $c_{\max}$ its maximum element. Given the condition $\frac{1}{\alpha}c_{\max} \leq \sum_{i=1}^{K} c_i^*$, one can always find $c^* = (c_1^*,\ldots,c_K^*)$ such that $\forall i \in (1,\ldots,K), c_i > c_i^*$. Indeed, we can set $c_1^* = \frac{1}{\alpha}(c_{\max} - \epsilon_1)$ for some $\epsilon_1 > 0$ and let $c_i^* = \frac{1}{\alpha}\left(\frac{\epsilon_1+\epsilon_2}{K-1}\right), \forall i \in [2,\ldots,K]$, where the value of $\epsilon_2 > 0$ can be made infinitely small. In this case, the condition $\frac{1}{\alpha}c_{\max} \leq \sum_{i=1}^{K} c_i^*$ is verified and the values ϵ_1 and ϵ_2 can be chosen to ensure that $\forall i \in [1,\ldots,K], c_i > c_i^*$ giving the desired result.

Finally, we note that the inequality established in this theorem holds even if agents are assumed to be self-interested and may lie about their true labels. This follows from the results established in [DFP10] where the authors prove that costs defined as an empirical risk of the best hypothesis consistent with the learning sample encourage the agents to tell the truth about their labels. This, however, is only the case when $\mathcal{H}$ is a space of constant or homogeneous linear functions over $\mathbb{R}^d$.

4.3 Collaborative Learning with Heterogeneous Sources

For our first result, we considered a setting where data distributions that generated individual samples of all agents are the same. This assumption, however, is often violated in practice as different data sources may provide data samples with important statistical differences. Consider the previous example with hospitals and assume now that the MRI scans are acquired on scanners with varying resolutions and sizes of the resulting images, thus leading to a shift in the statistical distribution between the different acquired samples. In this case, we fall into the category of the so-called transfer learning algorithms that may take the form of multi-task learning, domain adaptation or distributed learning problems. In what follows, we refer to any instance of a learning problem with heterogeneous samples following different probability distributions as to a general collaborative learning problem. In this scenario, a quantity of interest that one may want to quantify is the ratio between the number of samples that are needed to learn a good hypothesis for each agent and that required by a given collaborative algorithm to produce a hypothesis (or a set of hypotheses) that performs well on all of them. In the context of PAC learnability, this can be formalized as follows.

Definition 3. *Let $\mathcal{H}$ be a hypothesis class of VC dimension d. We say that a hypothesis $h \in \mathcal{H}$ allows to (ϵ,δ)-solve a learning problem $(\mathcal{H},\mathcal{D})$ if for any $\epsilon,\delta > 0$ with probability $1-\delta$ the following holds:*

$$\Pr_{\mathbf{x}\sim\mathcal{D}} \left[\mathbb{R}_{\mathcal{D}^i}(h(\mathbf{x}),y) \leq \epsilon\right] \geq 1-\delta.$$

Let us now define a general collaborative learning problem as a 2-tuple $(\mathcal{H},\{\mathcal{D}^i\}_{i=1}^K)$. We say that a hypothesis $h \in \mathcal{H}$ allows to (ϵ,δ)-solve a learning problem $(\mathcal{H},\{\mathcal{D}^i\}_{i=1}^K)$ if with probability $1-\delta$, $\mathbb{R}_{\mathcal{D}^i}(h(\mathbf{x}),y) \leq \epsilon$ for all $i \in [1,\ldots,K]$. We further define the sample complexity $m^i_{\epsilon,\delta}$ as the size of sample S^i drawn from $\mathcal{D}^i$ required by h to (ϵ,δ)-solve a problem $(\mathcal{H},\mathcal{D}^i)$. We assume that $\mathcal{H}$ is fixed for all individual agents so that, as shown in [AB09], the sample complexity becomes equal to $m^i_{\epsilon,\delta} = m_{\epsilon,\delta} = \mathcal{O}(\frac{1}{\epsilon}(d\log\left(\frac{1}{\epsilon}+\frac{1}{\delta}\right))$, $\forall i \in [1,\ldots,K]$. As an example of a collaborative learning algorithm, we consider [BHPQ17, Algorithm 2][2] and denote it by $\mathcal{L}$ in what follows.

In this setting, we can now state the following theorem.

Theorem 4. *Let $\mathcal{H}$ by a hypothesis space of VC dimension d. Let $m^*_{\epsilon,\delta}$ be a sample complexity required to (ϵ,δ)-solve the collaborative learning problem $(\mathcal{H},\{\mathcal{D}^i\}_{i=1}^K)$ by a hypothesis $h \in \mathcal{H}$ outputted by $\mathcal{L}$. Then, the following holds:*

$$\frac{m_{\epsilon,\delta}}{m^*_{\epsilon,\delta}} \leq \frac{\Theta(\log(K))}{K}.$$

Proof. Similar to the previous theorem, the idea behind our proof is to represent a collaborative learning problem $(\mathcal{H},\{\mathcal{D}^i\}_{i=1}^K)$ as a fair cost sharing game given in Figure 2. To this end, we let the nodes a_i correspond

[2] For the sake of completeness, we provide the description of this algorithm and the theoretical result related to it in the Supplementary material.

to the source nodes of agents $\{a_i\}_{i=1}^K$ with their respective distributions $\mathcal{D}^i$ and let the pooling node be the same as before. The destination node $\mathbf{L}$ is the state where every problem $(\mathcal{H}, \{\mathcal{D}^i\})$ is (ϵ, δ)-solved. The edge between node $\mathbf{P}$ and $\mathbf{L}$ corresponds to applying $\mathcal{L}$ on the received samples from node $\mathbf{P}$. Each agent has a choice between generating a sample of size $m_{\epsilon,\delta}$ to (ϵ, δ)-solve their problem or generating a sample for a collaborative learner $\mathcal{L}$. In the proposed setting, we can set the costs of individual edges $c_i = m^i_{\epsilon,\delta} = m_{\epsilon,\delta}$, $\forall i \in [1, \dots, K]$. The shared edge thus has a cost of $c^* = m^*_{\epsilon,\delta}$, where the $m^*_{\epsilon,\delta}$ examples are drawn from a uniform mixture distribution $\frac{1}{K}\sum_{i=1}^K \mathcal{D}^i$. In order to learn a hypothesis in a collaborative setting, we consider the algorithm presented in [BHPQ17, Algorithm 2] that has a property of (ϵ, δ)-solving a collaborative problem $(\mathcal{H}, \{\mathcal{D}^i\}_{i=1}^K)$ with $\frac{m^*_{\epsilon,\delta}}{m_{\epsilon,\delta}} = \mathcal{O}(\log^2(K))$ when $K = \mathcal{O}(d)$. In this case, agents $\{a_i\}_{i=1}^K$ tend to prefer paying a cost of $m_{\epsilon,\delta} \leq m^*_{\epsilon,\delta}$ for (ϵ, δ)-solving $(\mathcal{H}, \{\mathcal{D}^i\})$, $\forall i \in [1, \dots, K]$ thus leading to the Nash equilibrium with a cost equal to $Km_{\epsilon,\delta}$. However, the optimal solution is to learn in a collaborative way with a cost $m^*_{\epsilon,\delta} < Km_{\epsilon,\delta}$. As before, we can now bound the ratio between the Nash equilibrium and the optimal cost using Theorem 1 yielding the final result:

$$\frac{m_{\epsilon,\delta}}{m^*_{\epsilon,\delta}} \leq \frac{\Theta(\log(K))}{K}.$$

□

Remark 1. *Note that the established result can be proved in a more general setting without specifying the collaborative algorithm used to output a hypothesis. In this case, one would need to make an assumption similar to* (1) *that would restrict agents from choosing the shared path and ensure that the overall sample complexity of collaborative learning is smaller than the sum of sample complexities of each agent. This assumption, however, is almost always verified in practice as one can barely hope to find a collaborative algorithm that can* (ϵ, δ)*-solve* K *problems with a sample complexity smaller than that of each individual problem. The optimality condition,* $Km_{\epsilon,\delta} > m^*_{\epsilon,\delta}$*, is also far from being restrictive as it means that the chosen collaborative algorithm should perform at least a little bit better than a naive algorithm that takes a number of samples from each agent large enough to* (ϵ, δ)*-solve each problem* $(\mathcal{H}, \{\mathcal{D}^i\})$*.*

Remark 2. *Contrary to the first result obtained for homogeneous data sources, here we consider a simple fair cost sharing game with equal weights of all agents. As sample complexity remains the same for a fixed hypothesis space, it is reasonable to assume that agents participate equally in sharing the cost of the common edge. In the previous case, however, the sum of empirical errors defining the cost* c^* *consisted of terms that were not necessarily equal and thus the individual cost of each agent taking the shared path was allowed to be unequal too.*

It is import to underline that this result establishes a lower bound for the ratio between the collaborative and individual sample complexities (referred to as overhead) considered in [BHPQ17]. As mentioned in the proof, [BHPQ17, Theorem 3.1] states that an algorithm for collaborative learning proposed in their paper has a sample complexity that is only $\mathcal{O}(\log^2(K))$ larger than the individual sample complexity of solving each problem $(\mathcal{H}, \{\mathcal{D}^i\})$. Obviously, in this case it achieves an exponential improvement over a naive algorithm that takes a number of samples from each agent large enough to (ϵ, δ)-solve each problem $(\mathcal{H}, \{\mathcal{D}^i\})$. The two results can be further combined in a unified statement that takes the following form:

$$\frac{K}{\Theta(\log(K))} \leq \mathcal{O}\left(\log^2(K)\right) = \frac{m^*_{\epsilon,\delta}}{m_{\epsilon,\delta}}.$$

Both this unification and the previous theorem highlight the practical interest in expressing various learning problems as instances of fair cost sharing game as it allows to provide an analysis of learning settings in a completely different, and arguably simpler, way.

5 Conclusions and Future Perspectives

In this paper, we considered collaborative learning problem widely presented in machine learning as an instance of fair cost sharing games. In order to relate the two, we showed how a collaborative learning scenario can be represented as a graph of a fair cost sharing game. The established link allowed us to analyze collaborative learning problem in two settings where the learning samples available to agents were assumed to be drawn from the same or different probability distributions. For these two cases, we proved two distinct theoretical results where the first one bounds the ratio between the optimal empirical risk of non-collaborative and collaborative learning approaches, while the second establishes the same kind of bound for their respective sample complexities. The proposed analysis

naturally completes a previous work on the subject and brings a new point of view for the general collaborative learning scenario. To the best of our knowledge, there were no other attempts in analyzing this particular learning setting using the results established for fair cost sharing games in game theory.

Due to the rich body of literature on fair cost sharing games that exist in algorithmic game theory, the possible future perspectives of this work are many. Among them, one of the most important ones is to consider a fair cost sharing game where pooling the data has a certain cost for all agents that is different from 0. Indeed, in many real-world situations, agents represented by industrial actors may want to protect their data from its disclosure either to the learner or to the other agents. This conflict of interests is often modeled as a privacy-aware learning paradigm where the optimized objective function considers two terms representing the trade-off between the privacy of the data used for learning and the optimal learning itself. Fair cost sharing games offer a very intuitive way of modeling the abovementioned situation by the means of imposing a nonzero cost on the edges that allow the agents to pool data together. In this case, the analysis proposed in this paper may be extended to study the possible gain of collaborative learning with privacy constraints and can be done, for instance, by using a traditional framework of differential privacy [Dwo06] and PAC learnability results related to it [SPKS11, DR14]. On the other hand, this setup can be also potentially applied to model the multi-source domain adaptation problem where the source nodes correspond to the available source domains, while the destination node is given by the target domain. In this case, the direct link between source nodes and the target one may correspond to the error achieved by a direct application of a source classifier to the target data usually considered as a weak baseline in domain adaptation. On the contrary, the shared edge would correspond to minimizing the combined weighted error over all sources in the spirit of [BCK+07] so that the cost would be defined based on the generalization bounds established by the authors of this paper. Both these research lines merit a thorough future investigation as they aim to tackle two important drawbacks of learning phenomenon such as its lack of generalizing capacity across different domains and the rising concerns regarding the privacy preservation.

References

[AB09] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations.* Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[ADK+04] Elliot Anshelevich, Anirban Dasgupta, Jon Kleinberg, Eva Tardos, Tom Wexler, and Tim Roughgarden. The price of stability for network design with fair cost allocation. In *FOCS*, pages 295–304, 2004.

[ADTW03] Elliot Anshelevich, Anirban Dasgupta, Eva Tardos, and Tom Wexler. Near-optimal network design with selfish agents. In *STOC*, pages 511–520, 2003.

[Bax97] Jonathan Baxter. A bayesian/information theoretic model of learning to learn viamultiple task sampling. *Machine Learning*, 28(1):7–39, 1997.

[BBFM12] Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. In *COLT*, pages 26.1–26.22, 2012.

[BBHM05] M. F. Balcan, A. Blum, J. D. Hartline, and Y. Mansour. Mechanism design via machine learning. In *FOCS*, pages 605–614, 2005.

[BCK+07] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning bounds for domain adaptation. In *NIPS*, pages 129–136, 2007.

[BDBC+10] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[BH05] Avrim Blum and Jason D. Hartline. Near-optimal online auctions. In *SODA*, pages 1156–1163, 2005.

[BHPQ17] Avrim Blum, Nika Haghtalab, Ariel D Procaccia, and Mingda Qiao. Collaborative pac learning. In *NIPS*, pages 2392–2401, 2017.

[BKRW03] Avrim Blum, Vijay Kumar, Atri Rudra, and Felix Wu. Online learning in online auctions. In *SODA*, pages 202–204, 2003.

[BPZ15] Maria-Florina Balcan, Ariel D. Procaccia, and Yair Zick. Learning cooperative games. In *IJCAI*, pages 475–481, 2015.

[BS11] Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *ACM SIGKDD*, pages 547–555, 2011.

[BSV17] Eric Balkanski, Umar Syed, and Sergei Vassilvitskii. Statistical cost sharing. In *NIPS*, pages 6221–6230, 2017.

[Car97] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[CB98] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI*, pages 746–752, 1998.

[CR09] Ho-Lin Chen and Tim Roughgarden. Network design with weighted players. *Theoretical Computer Science*, 45(2):302–324, 2009.

[DFP10] Ofer Dekel, Felix Fischer, and Ariel D. Procaccia. Incentive compatible regression learning. *Journal of Computer and System Sciences*, 76(8):759–777, 2010.

[DLM17] L. Dritsoula, P. Loiseau, and J. Musacchio. A game-theoretic analysis of adversarial classification. *IEEE Transactions on Information Forensics and Security*, 12(12):3094–3109, 2017.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3):211–407, 2014.

[Dwo06] Cynthia Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[FS99] Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[HSE95] Shai Herzog, Scott Shenker, and Deborah Estrin. Sharing the cost of multicast trees: An axiomatic analysis. *SIGCOMM Comput. Commun. Rev.*, 25(4):315–327, 1995.

[HW03] Junling Hu and Michael P. Wellman. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[KI12] Abhishek Kumar and Hal Daumé III. Learning task grouping and overlap in multi-task learning. In *ICML*, page 1103–1110, 2012.

[LC09] W. Liu and S. Chawla. A game theoretical model for adversarial learning. In *ICDM Workshops*, pages 25–30, 2009.

[LCQ15] Tie-Yan Liu, Wei Chen, and Tao Qin. Mechanism learning with mechanism induced data. In *AAAI*, pages 4037–4041, 2015.

[MMR09a] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009.

[MMR09b] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Multiple source adaptation and the rényi divergence. In *UAI*, pages 367–374, 2009.

[MPR12] Reshef Meir, Ariel D. Procaccia, and Jeffrey S. Rosenschein. Algorithms for strategyproof classification. *Artificial Intelligence*, 186:123–156, 2012.

[Nas50] J. F. Nash. Equilibrium points in n-person games. *National Academy of Sciences of the USA*, 36(48-49), 1950.

[Nas51] J. F. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[PKMK00] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. Learning to cooperate via policy search. In *UAI*, pages 489–496, 2000.

[PL05] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[Sha53] L. S. Shapley. A value for n-person games. In *Contributions to the Theory of Games*, volume II, page 307–317. Princeton University Press, 1953.

[SPKS11] Kobbi Nissim Sofya Raskhodnikova Shiva Prasad Kasiviswanathan, Homin K. Lee and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[SSS07a] Shai Shalev-Shwartz and Yoram Singer. Convex repeated games and fenchel duality. In *NIPS*, pages 1265–1272, 2007.

[SSS07b] Shai Shalev-Shwartz and Yoram Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 69(2-3):115–142, 2007.

[SZ16] Dale Schuurmans and Martin A Zinkevich. Deep learning games. In *NIPS*, pages 1678–1686, 2016.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[WKS16] Jialei Wang, Mladen Kolar, and Nathan Srerbo. Distributed multi-task learning. In *AISTATS*, pages 751–760, 2016.

Classification de séries temporelles basée sur des "shapelets" interprétables par réseaux de neurones antagonistes

Yichang Wang[1], Rémi Emonet[2], Elisa Fromont[1], Simon Malinowski[1], Etienne Menager[3], Loïc Mosser[3], et Romain Tavenard[4]

[1]Univ. Rennes 1 - IRISA/INRIA rba
[2]Univ. Lyon - Laboratoire Hubert Curien, UMR 5516
[3]ENS Rennes - IRISA/INRIA rba
[4]Univ. Rennes 2 - LETG

Abstract

Times series classification can be successfully tackled by jointly learning a shapelet-based representation of the series in the dataset and classifying the series according to this representation. However, although the learned shapelets are discriminative, they are not always similar to pieces of a real series in the dataset. This makes it difficult to interpret the decision, i.e. difficult to analyze if there are particular behaviors in a series that triggered the decision. In this paper, we make use of a simple convolutional network to tackle the time series classification task and we introduce an adversarial regularization to constrain the model to learn more interpretable shapelets. Our classification results on all the usual time series benchmarks are comparable with the results obtained by similar state-of-the-art algorithms but our adversarially regularized method learns shapelets that are, by design, interpretable.

Mots-clef: Time series, Shapelets, Adversarial networks, Interpretable models, Convolutional neural networks.

1 Introduction

A time series (TS) Z is a series of time-ordered values, $Z = z^{(1)}, z^{(2)}, \ldots, z^{(T)}$ where $z^{(t)} \in R^d$, T is the length of our time series and d is the dimension of the feature vector describing each data point. If $d = 1$, Z is said univariate, otherwise it is said multivariate. In this paper, we are interested in the Time Series Classification (TSC) task. We are given a training set $\mathcal{T} = \{(Z_1, y_1), \ldots, (Z_n, y_n))\}$, composed of n time series Z_i and their associated labels y_i (target variable).

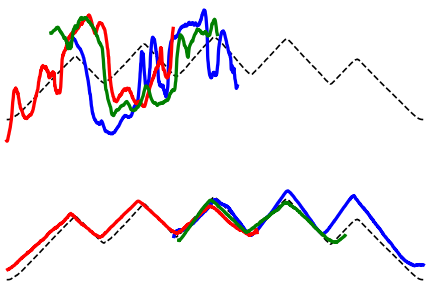


Figure 1: Example test time series and three most discriminative shapelets used for its classification for a baseline [GSWST14] (top) and for our proposed AI↔PR-CNN model (bottom) on the HandOutlines classification problem.

Our aim is to learn a function h such that $h(Z_i) = y_i$, in order to predict the labels of new incoming time series. The time series classification problem has been studied in countless applications (see for example [SS05]) ranging from stock exchange evolution, daily energy consumptions, medical sensors, videos, etc.

Many methods have been developed to tackle this problem (see [BLB+17] for a review). One very successful category of methods consists in jointly learning a shapelet-based representation of the series in the dataset and classifying the series according to this representation [LDHB12, GSWST14]. Shapelets are discriminative phase-independent subsequences. An example of such a shapelet is given in Figure 1. However, if the learned shapelets are definitively discriminative, they are often different from actual pieces of a real series

in the dataset. As such, the classification decision is difficult to interpret, i.e. it is difficult to determine what particular behavior in a time series triggered the classification decision. Note that the same interpretability issue arrises with ensemble classifiers such as [BLHB15] where one decision depends on the presence of multiple shapelets. One of the main challenge nowadays is to enrich Machine Learning (ML) systems, and in particular black box models such as neural networks, so that they have the ability to explain their outputs to human users. In many scenarios, it may be risky, unacceptable, or simply illegal, to let artificial intelligent systems make decisions without any human supervision [GMR+18]. Hence, it is necessary for ML systems to provide an explanation of their decisions to all the humans concerned.

In this paper, we make use of a simple convolutional network to classify time series and we show how one can use adversarial techniques to regularize the parameters of this network such that it learns more interpretable shapelets. Section 2 presents the related work on time series classification, interpretability of models and adversarial training. We present our adversarial parameter regularization method in Section 3. In Section 4, we show quantitive and qualitative results on the usual time series benchmarks [BLVK] that are both in pair with state-of-the-art methods and very interesting to interpret the neural network predictions.

2 Related Work

In this section we review the literature on Time Series Classification (TSC), on tools for understanding black box model predictions and on adversarial training.

2.1 Time Series Classification

In the TSC literature, two main families of approaches have been designed. First, a dedicated metric can be used to compare the time series. In this case the decision is based on the resulting similarities. For example, [SC78] uses Dynamic Time Warping (DTW) to find an optimal alignment between time series and provides an alignment cost that can be used to assess the similarity. Another family of methods is based on the extraction of features in the time series. Among these works, shapelet-based classifiers have attracted a lot of attention from the research community.

Shapelets are discriminative subseries that can either be extracted from a set of time series or learned so as to minimize an objective function. They have been introduced in [YK09], in which a binary decision tree is built, whose nodes are shapelets and whose subtrees contain subsets of time series that contain or not that shapelet. In this work, shapelets are extracted from a training set of time series and building the decision tree requires to test all possible subseries from the training set, which makes the method intractable for large-scale learning with an overall time complexity of $O(n^2 \cdot T^3)$ where n is the number of training time series and T is the length of time series in the training set. This high time complexity has led to the use of heuristics in order to select the shapelets more efficiently. In [RK13] (Fast Shapelets), the authors rely on quantized time series and random projections in order to fasten the shapelet search. Note however that these improvements in time complexity are obtained at the cost of a lower classification accuracy, as reported in [BLB+17]. The Shapelet Transform (ST) [LDHB12] consists in transforming time series into a feature vector whose coordinates represent distances between the time series and the shapelets selected beforehand. It hence needs to select a shapelet set (as in [YK09]) before transforming the time series. The resulting vectors are then given to a classifier in order to build the decision function. Training time complexity for ST is in $O(n^2 \cdot T^4)$ [FFW+18], which makes it unfit for large scale learning.

In order to face the high complexity that comes with search-based methods, other strategies have been designed for shapelet selection. On the one hand, some attention has been paid to random sampling of shapelets from the training set [KPB16]. On the other hand, Grabocka *et al.* [GSWST14] showed that shapelets could be learned using a gradient-descent-based optimization algorithm. The method, referred to as Learning Shapelets (LS) in the following, jointly learns shapelets and parameters of a logistic regression classifier, which makes it very similar in spirit to a neural network with a single convolutional layer followed by a fully connected classification layer and where the convolution operation is replaced by a sliding-window local distance computation. A min-pooling aggregator is used for temporal aggregation.

Closely related to shapelet-based methods (as stated above), variants of Convolutional Neural Networks (CNN) have been introduced for the TSC task [WYO17]. These are mostly mono-dimensional variants of CNN models developed in the Computer Vision field. Note however that most models are rather shallow, which is likely to be related to the moderate sizes of the benchmark datasets present in the UCR/UEA archive [BLVK]. A review of these models can be found in [FFW+18].

Finally, ensemble-based methods, such as COTE [BLHB15] or HIVE-COTE [LTB18], that rely on several of the above-presented standalone classifiers are now considered state-of-the-art for the TSC task. Note however that these methods tend to be computationally expensive, with high memory usage and difficult to interpret (as stated in the Section 1) due to the combination of many different core classifiers.

2.2 Model Interpretability

Among the vast number of existing classifiers, some are easily interpretable (e.g. decision trees, classification rules), while others are difficult to interpret (e.g. ensemble methods, neural networks that can be considered as black-box). In this paper we are interested in understanding the decisions of a neural network, i.e. of a black-box classifier. Interpretation of black box classifiers usually consists in designing an interpretation layer between the classifier and the human level. Two criteria refine the category of methods to interpret classifiers: global versus local explanations, and black-box dependent versus agnostic. In this category, state-of-the-art methods are Local Interpretable Model-agnostic Explanations (LIME and Anchors) [RSG16, RSG18] and SHapley Additive exPlanations (SHAP) [LL17]. SHAP values come with the black-box local estimation advantages of LIME, but also with theoretical guarantees. A higher absolute SHAP value of an attribute compared to another means that it has a higher predictive or discriminative power.

For our neural network-based TSC model, we believe that the explanations should not directly come from the vector of attributes describing each point of each time series but from the discriminative shapelets learned as an intermediate representation to classify the series. Solutions such as LIME, Anchors and SHAP which are not designed to inspect the internal representation of a model are thus not well suited for our problem.

Fang *et al.* [FWW18] have a similar goal as ours (to produce interpretable discriminative shapelets) and build on both the works from [LDHB12] (in this case the candidate shapelets are extracted with a piecewise aggregate approximation) and from [GSWST14] to automatically refine the "handcrafted" shapelets. Contrarily to our method, there is no explicit constraint on the learning process that ensures the interpretability of the shapelets. Besides, their experimental validation makes it hard to fully grasp the benefits and limitations of the proposed method since the algorithm is evaluated on a small subset of UCR/UEA datasets [BLVK] and they provide visualizations for only a couple of the learned shapelets.

2.3 Adversarial Training

Adversarial training of neural networks has been popularized by Generative Adversarial Networks (GANs) [GPM+14] and their numerous variants.[1] A GAN is a combination of two neural networks: a generator and a discriminator which compete against each other during the training process to reach an equilibrium where the discriminator cannot distinguish between the generator outputs and real unlabelled training data. In a GAN, the adversarial network is used to push the generator towards producing data as similar to real data as possible. Other (non generative) adversarial training settings have been studied, for example in the context of domain adaptation [THSD17]. In this case, the adversarial network is used to regularize the latent representation learned by the classifier such that it becomes domain-independent. The recent work from [ZKZ+18] also uses adversarial regularization to constrain the latent representation of an autoencoder to follow a given distribution. This idea is different from what we propose in this article as 1) we use a non-generative supervised adversarial approach, 2) we use it in a different context and 3) we do not regularize the latent representation but directly the convolution parameters of the network so that they are similar to real subseries from the training data.

3 Proposed Method

In this section, we present our approach to learn interpretable discriminative shapelets for time series classification.

Our base time series *classifier* is a Convolutional Neural Network (CNN). As explained in Section 2, this model is very similar in spirit to the Learning Shapelet (LS) model presented in [GSWST14].

Both LS and CNN slide the shapelets on the series to compute local (dis)similarities. The main difference between the classifier of LS and that of our method is the (dis)similarity between a shapelet and a series. LS uses a squared Euclidean distance between a portion of the time series Z starting at index i and a shapelet S of length L:

$$D(z_{i:i+L}, S) = \sum_{l=1}^{L} \left(z^{(i+l-1)} - S^{(l)} \right)^2 . \qquad (1)$$

[1] See `https://github.com/hindupuravinash/the-gan-zoo` for a list.

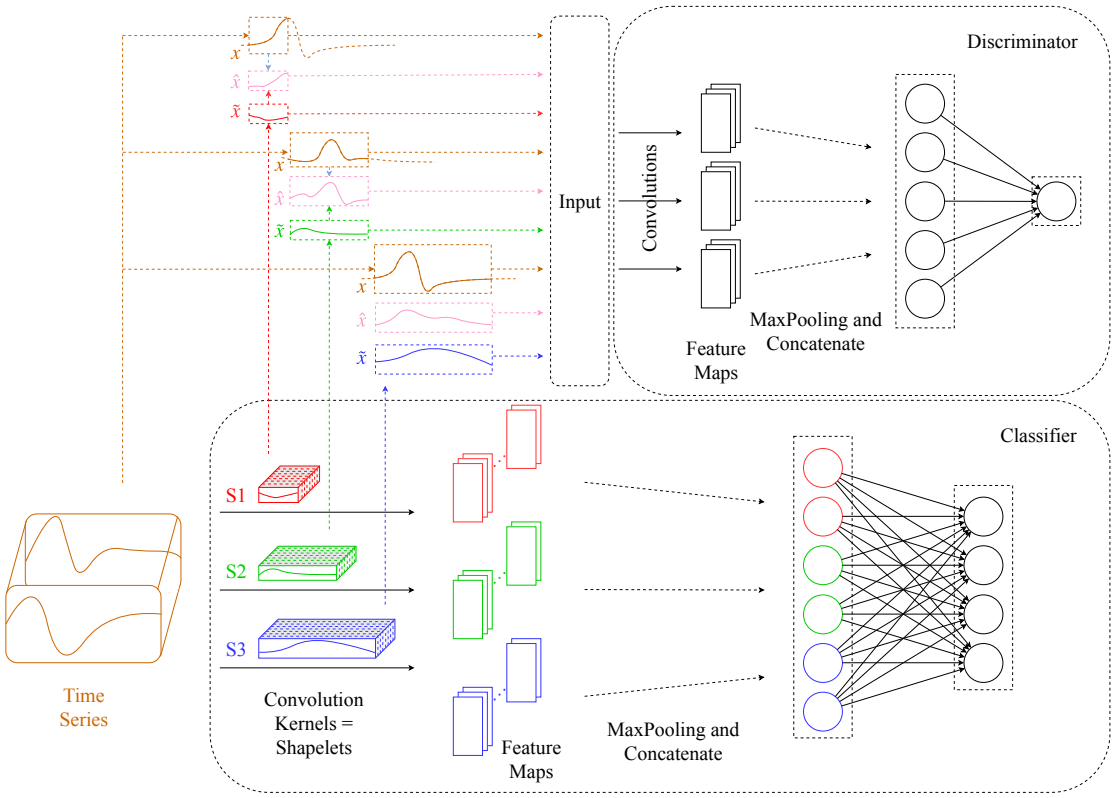


Figure 2: Architecture of our proposed Adversarially Input↔Parameter Regularized CNN (AI↔PR-CNN)

The smaller this distance, the closer the shapelet is to the considered subseries. In a CNN, the feature map is obtained from a convolution, and hence encodes cross-correlation between a series and a shapelet:

$$D(z_{i:i+L}, S) = \sum_{l=1}^{L} z^{(i+l-1)} \cdot S^{(l)}. \quad (2)$$

Note that here, the higher $D(z_{i:i+L}, S)$, the more similar the shapelet is to the subseries.

As shown in Figure 2 (bottom), the convolutional layer of this classifier is made of three parallel convolutional blocks with shapelets of different lengths (red, green, blue) to be comparable with the structure proposed in LS. We will loosely refer to the convolution filters of our classifier as *Shapelets* in the following.

Inspired by previous works on adversarial training (see e.g. Section 2), in addition to our CNN classifier, we make use of an adversarial neural network (the discriminator at the top of Figure 2) to regularize the convolution parameters of our classifier. This regularization acts as a soft constraint for the classifier to learn shapelets as similar to real pieces of the training time series as possible.

This novel regularization strategy is referred to, in the following, as Adversarial Input↔Parameter Regularization (AI↔PR) and the corresponding model is named AI↔PR-CNN.

Contrarily to GANs, our adversarial architecture does not rely on a generator to produce fake samples from a latent space. The AI↔PR strategy iteratively modifies the shapelets (i.e. the convolution filters of the classifier) such that they become close to subseries from the training set. To execute this strategy, the discriminator is trained to distinguish between real subseries from the training set and the shapelets. During the regularization phase, the discriminator updates the shapelets so that they become more and more similar to real subseries.

To obtain the best trade-off between the discriminative power of the shapelets (i.e. the final classification performance) and their interpretability, our training procedure alternates between training the discriminator

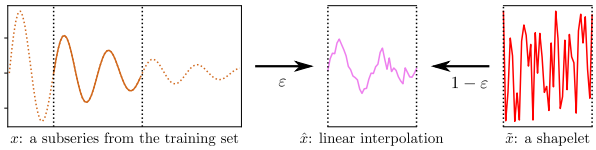


Figure 3: An example of samples provided as input to the discriminator.

and the classifier.

The type of data given as input to the discriminator is another major difference between a GAN and AI↔PR-CNN: in a GAN, the discriminator is fed with complete instances, while in AI↔PR-CNN, the discriminator takes subseries as input. These subseries can either be shapelets from the classifier model (denoted as $\tilde{x}$ in Figure 2), portions of training time series (denoted as x) or interpolations between shapelets and training time series portions ($\hat{x}$, see the following section for more details on those), as illustrated in Figure 3. This process allows the discriminator to alter the shapelets for better interpretability.

3.1 Loss function

As for GANs, our optimization process alternates between losses attached to the subparts of our AI↔PR-CNN model. Here, each training epoch consists of three main steps that are (i) optimizing the discriminator parameters for correct classification, (ii) optimizing the discriminator parameters to better distinguish between real subseries and shapelets and (iii) optimizing shapelets to fool the discriminator. Each of these steps is attached to a loss function that we describe in the following.

Firstly, a multi-class cross entropy loss is used for the classifier. It is denoted by $L_c(\theta_c)$ where θ_c is the set of all classifier parameters.

Secondly, our discriminator is trained using a loss function derived from the Wasserstein GAN with Gradient Penalty (WGAN-GP) [GAA+17]:

$$L_d(\theta_d) = \mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_S}[D(\tilde{x})] - \mathop{\mathbb{E}}_{x\sim\mathbb{P}_x}[D(x)] + \lambda \mathop{\mathbb{E}}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\left[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2\right] \quad (3)$$

where $\mathbb{P}_S$ is the empirical distribution over the shapelets, $\mathbb{P}_x$ is the empirical distribution over the training subseries, and

$$\hat{x} = \epsilon x + (1-\epsilon)\tilde{x}, \quad (4)$$

where ϵ is drawn uniformly at random from the interval $[0, 1]$ (cf. Figure 3) .

Thirdly, shapelets are updated to fool the discriminator by optimizing on the loss $L_r(\theta_s)$ where $\theta_s \subset \theta_c$ is the set of shapelet coefficients:

$$L_r(\theta_s) = -\mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_S}[D(\tilde{x})]. \quad (5)$$

3.2 Learning Algorithm

Algorithm 1 presents the whole training procedure to update the parameters of our AI↔PR-CNN model. At each epoch of this algorithm, the three steps presented above are executed sequentially. Note that in the second step (lines 10–17), sampling classifier shapelets, as well as sampling subseries from the training set, is performed uniformly at random.

4 Experiments

In this section, we will detail the training procedure for the AI↔PR-CNN and present both quantitative and qualitative experimental results.

4.1 Experimental Setting

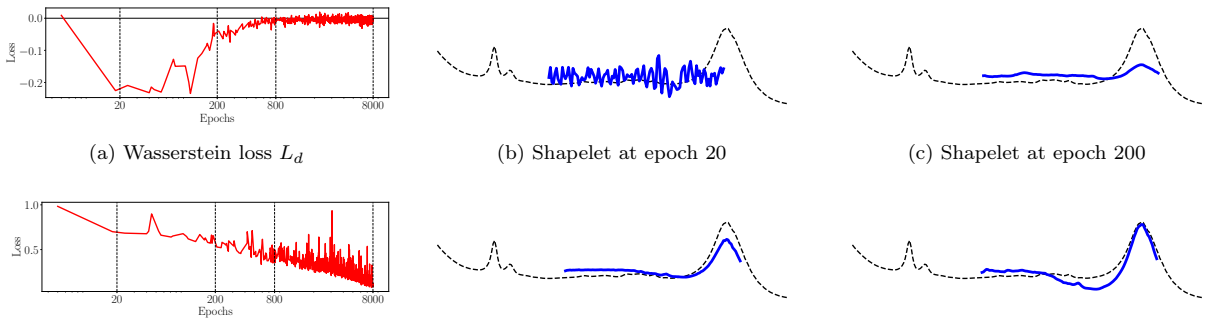
As explained in Section 2, our most relevant competitor is Learning Shapelets (LS) from [GSWST14] as it also describes a shapelet-based model where the shapelets are learned and where a single model is used for classification. In the following sections, all the results presented for LS are retrieved from the UCR/UEA repository [BLVK] and the shapelets presented for LS are obtained using tslearn implementation [Tav17].

4.1.1 Datasets

To compare our proposed method with [GSWST14], we use a set of 84 univariate time series datasets from the UCR/UEA repository [BLVK]. Note that our CNN-based method is not, by design, limited to univariate time series. However, for a fair comparison, we limited ourself to these datasets for this study. The datasets are significantly different from one to another, including seven types of data with various number of instances, lengths, and classes. The splits between training test sets are provided.

4.1.2 Architecture details and parameter setting

We have implemented the AI↔PR-CNN model using TensorFlow [AAB+15] following the general architecture illustrated in Figure 2. The classifier is composed



(a) Wasserstein loss L_d (b) Shapelet at epoch 20 (c) Shapelet at epoch 200

(d) Cross-entropy loss L_c (e) Shapelet at epoch 800 (f) Shapelet at epoch 8000

Figure 4: Illustration of the evolution of a shapelet during training (for the Wine dataset).

of one 1D convolution layer with ReLU activation, followed by a maxpooling layer along the temporal dimension and a fully connected layer with softmax activation. The shapelets use a Glorot uniform initializer [GB10] while other weights are initialized uniformly (using a fixed range). For each dataset, three different shapelet lengths are considered, inspired by the heuristic from [GSWST14] but without resorting to hyper-parameter search: we consider 3 groups of $20 \times n_{\text{classes}}$ shapelets of length $0.2T$, $0.4T$ and $0.6T$, where n_{classes} is the number of classes in the dataset and T is the length of the time series at stake.

The convolution filters of the classifier, i.e. the shapelets, are given as input to the discriminator which has the same structure as the classifier, but with shorter convolution filters (100 filters of size $0.06T$, $0.12T$ and $0.18T$) and a single-neuron *tanh* activation instead of the softmax in the last layer. For optimization, we use Adam optimizer with a standard parametrization and each epoch consists in $n_c = 15$ (resp. $n_d = 20$ and $n_r = 17$) mini-batches of optimization for the classifier loss (resp. discriminator and regularizer losses).

Experimental results are reported in terms of test accuracy and aggregated over five random initializations. All experiments are run for 8000 training epochs.

4.2 Qualitative results

Our method aims at producing interpretable results in the sense that shapelets should be similar to sub-parts of some series from the dataset. We first validate that our AI↔PR scheme actually ensures that shapelets are similar to the training data.

We first illustrate our training process and its impact on a single shapelet in Figure 4. In this figure, we show the evolution of a given shapelet for the Wine dataset at epochs 20, 200, 800 and 8,000. One can see from the loss values reported in Figures 4a and 4d that these correspond to different stages in our learning process. At epoch 20, the Wasserstein loss is far from the 0 value ($L_d = 0$ corresponds to a case where the discriminator cannot distinguish between shapelets and real subseries), and this indeed corresponds to a shapelet that looks very different from an actual subseries. As epochs go, both the Wasserstein loss L_d and the cross-entropy one L_c get closer to 0, leading to both realistic and discriminative shapelets.

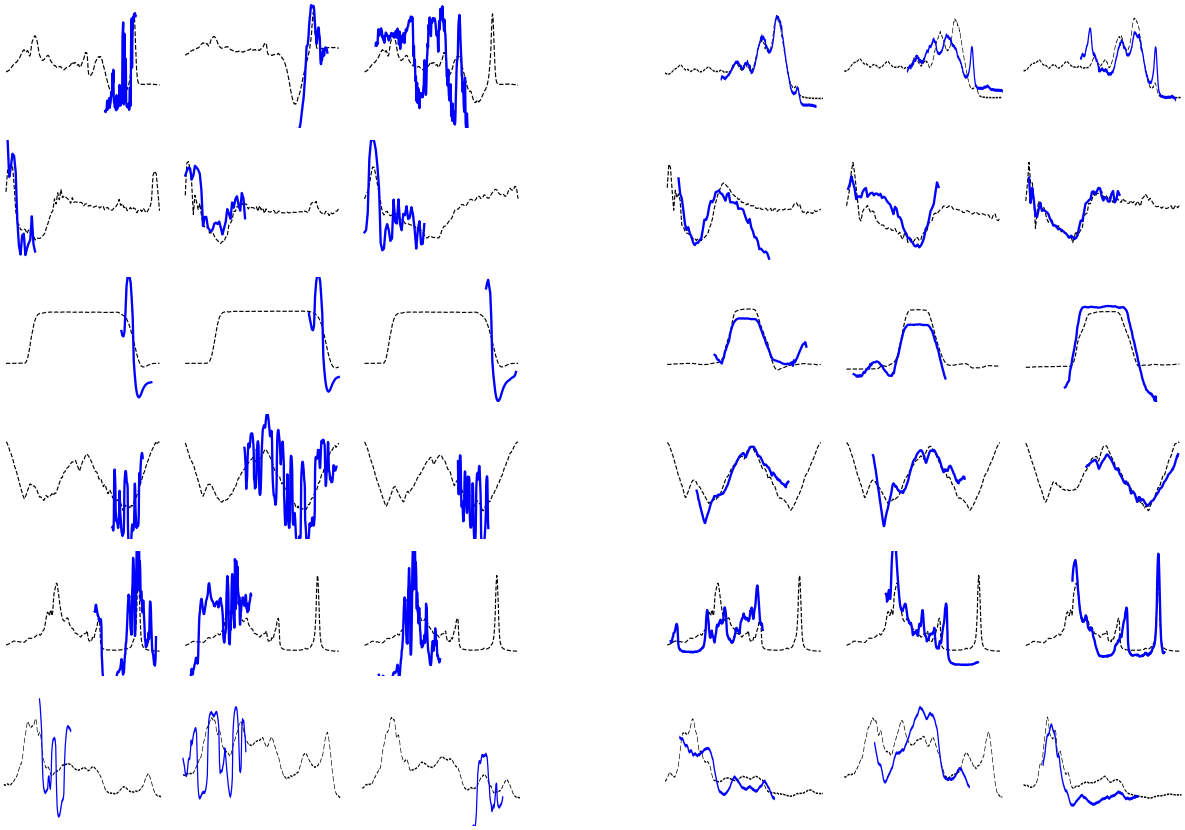
To further check the effect of our regularization, we focus on the most discriminative shapelets for a bunch of datasets, as it would be misleading to look at a random shapelet: a shapelet might well be similar to a series but useless for the classification. The discriminative power, for class j, of the shapelet at index s with respect to the i-th time series in the training set is evaluated as:

$$\mathcal{P}_{i,j}(s) = a(Z_i)_s \cdot w_{s,j} \tag{6}$$

where $a(Z_i)_s$ is the s-th component (i.e. the one that corresponds to shapelet s) of the activation map for the time series Z_i and $w_{s,j}$ is the weight connecting that s-th component to the j-th output in the logistic layer of our classifier. As we aim at evaluating the overall discriminative power of a shapelet in a multi-class setting, and given that we use of a soft-max activation at the output of our logistic layer, we can define the cross-class discriminative power of a shapelet as:

$$\mathcal{P}_i(s) = \sum_j \left(a(Z_i)_s \cdot (w_{s,j} - \max_{j'} w_{s,j'}) \right)^2 . \tag{7}$$

This is the criterion that we use to rank our shapelets in terms of discriminative power and to select the three



(a) Learning Shapelets [GSWST14]

(b) AI↔PR-CNN

Figure 5: The three most discriminative shapelets obtained for the datasets Beef, ECG200, GunPoint, Herring, OliveOil and Strawberry (rows 1 to 6, respectively) using (a) Learning Shapelets or (b) our AI↔PR-CNN architecture. The average discriminative power of the shapelets is evaluated using Eq. 7 and each shapelet is superimposed over its best matching time series in the training set.

most discriminative shapelets in Figure 5. This figure shows a significant improvement in terms of adequation of the shapelets to the training time series when using our AI↔PR-CNN model in place of a standard LS one.

Another important aspect in terms of interpretability is the explanation that can be provided to an end-user to explain a classification decision. Example of such settings are presented in Figure 1 in which, for a given test time series, we present the three shapelets that were the most important to make a classification decision. One can notice that in both cases, shapelets extracted by AI↔PR-CNN better fit the time series at stake and hence help the end-user focus on the particular pattern in the time series that leads to the decision (e.g. the series of three peaks for HandOutlines or the overall shape of the central hump for Herring).

4.3 Quantitative results

Our AI↔PR is able to recover shapelets that are discriminative and similar to the input, as expected. We want to quantify if this is achieved at the expense of classification accuracy. To do so, we analyze the accuracy obtained by three methods on the 84 datasets using scatter plots. We compare LS accuracy results with our AI↔PR-CNN method in Figure 7. In each figure, points on the diagonal are datasets for which the accuracy is similar for both competitors. We can see that, in terms of test accuracy, AI↔PR-CNN gives results comparable to LS. For most datasets, the differ-

Algorithm 1: Learning Interpretable Shapelet

Require : number of shapelets n_S
Require : random initialization for the classifier/discriminator/shapelets $\theta_c, \theta_d, \theta_s \subset \theta_c$
Require : gradient penalty coefficient λ
Require : number of epochs n_{epochs}, mini-batch size m
Require : number of classifier/discriminator/regularization mini-batches per epoch n_c, n_d, n_r
Require : optimizer (Adam) hyperparameters α, β_1, β_2

for $i = 1, \ldots, n_{epochs}$ **do**
  for $t = 1, \ldots, n_c$ **do**
    for $j = 1, \ldots, m$ **do**
      Sample a pair (Z_j, y_j) from the training set
      $\hat{y}_j \leftarrow h_{\theta_c}(Z_j)$
      $L_c^{(j)} \leftarrow \text{CrossEntropy}(y_j, \hat{y}_j)$
    end
    $\theta_c \leftarrow \text{Adam}(\nabla_{\theta_c} \frac{1}{m} \sum_{j=1}^{m} L_c^{(j)}, \theta_c, \alpha, \beta_1, \beta_2)$
  end
  for $t = 1, ..., n_d$ **do**
    for $j = 1, ..., m$ **do**
      Sample a shapelet $\tilde{x}_j$ from the set θ_s, a subseries x_j from the training set and a random number $\epsilon \sim U[0, 1]$
      $\hat{x}_j \leftarrow \epsilon x_j + (1 - \epsilon)\tilde{x}_j$
      $L_d^{(j)} \leftarrow D(\tilde{x}_j) - D(x_j) + \lambda(||\nabla_{\hat{x}} D(\hat{x}_j)||_2 - 1)^2$
    end
    $\theta_d \leftarrow \text{Adam}(\nabla_{\theta_d} \frac{1}{m} \sum_{j=1}^{m} L_d^{(j)}, \theta_d, \alpha, \beta_1, \beta_2)$
  end
  for $t = 1, \ldots, n_r$ **do**
    for $j = 1, \ldots, n_S$ **do**
      $\tilde{x}_j \leftarrow \theta_s[j]$
      $L_r^{(j)} \leftarrow -D(\tilde{x}_j)$
    end
    $\theta_s \leftarrow \text{Adam}(\nabla_{\theta_s} \frac{1}{n_S} \sum_{j=1}^{n_S} L_r^{(j)}, \theta_s, \alpha, \beta_1, \beta_2)$
  end
end

ence in accuracy is low, with a small edge for LS: on average for the 84 datasets, LS obtains an accuracy of 0.77 whereas AI↔PR-CNN obtains an accuracy 0.76. On three datasets (namely HandOutlines, NonInvasive-FetalECGThorax1 and OliveOil), our AI↔PR and its

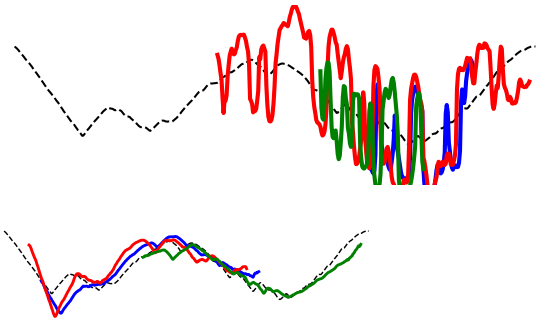


Figure 6: A test time series from the Herring dataset together with the three shapelets that were prominent for the classification decision. The results for LS are at the top, for our AI↔PR-CNN model a the bottom.

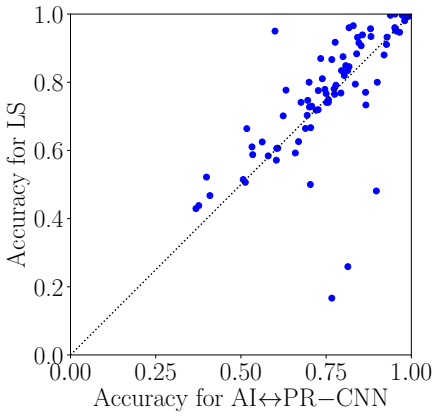


Figure 7: Accuracy comparison between Learning Shapelets and our AI↔PR method on 84 datasets (each point is a dataset) of the UCR/UEA repository [BLVK].

regularization seems to be strongly positive (and detrimental on one dataset), in terms of generalization. The simple CNN seems to give slightly better results than LS (and thus than our AI↔PR-CNN): on average for the 84 datasets, the simple CNN obtains an accuracy of 0.8 which means that our backbone neural network architecture is a good candidate to jointly learn shapelets and classify time series.

5 Conclusion

We have presented a new shapelet-based time series classification method that produces interpretable shapelets. The shapelets are deemed interpretable because they

are similar to pieces of a real series and can thus be used to explain a particular model prediction. The method is based on an adversarial architecture where one convolutional neural network is used to classify the series and another one is used to constrain the first network to learn interpretable shapelets. Our results show that the expected trade-off between accuracy and interpretability is satisfactory: our classification results are comparable with similar state-of-the-art methods while our shapelets are interpretable.

We believe that the proposed adversarial regularization method could be used in many more applications where the regularization should be put on the parameters instead of the latent representation of the networks as done, for example, with Generative Adversarial Networks.

In future work, we would like to first investigate the use of an additional regularization term, based on the group lasso [BEFO], to be able to determine automatically a minimal set of necessary interpretable shapeletes. We also want to use our regularization on other types of data (such as multivariate time series, spatial data, graphs) and in a deep CNN. Furthermore, we would like to adapt this architecture for unsupervised anomaly detection in time series with interpretable clues using neural network architectures such as convolutional auto-encoders or generative networks.

References

[AAB+15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[BEFO] K. Bascol, R. Emonet, E. Fromont, and J.-M. Odobez. Unsupervised Interpretable Pattern Discovery in Time Series Using Autoencoders. In A. Robles-Kelly, M. Loog, B. Biggio, F. Escolano, and R. Wilson, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 10029, pages 427–438. Springer International Publishing.

[BLB+17] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, May 2017.

[BLHB15] A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with cote: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.

[BLVK] A. Bagnall, J. Lines, W. Vickers, and E. Keogh. The uea & ucr time series classification repository. `www.timeseriesclassification.com`.

[FFW+18] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller. Deep learning for time series classification: a review. *ArXiv*, abs/1809.04356, 2018.

[FWW18] Z. Fang, P. Wang, and W. Wang. Efficient learning interpretable shapelets for accurate time series classification. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 497–508. IEEE, 2018.

[GAA+17] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[GB10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[GMR+18] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Survey*, 51(5), 2018.

[GPM+14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[GSWST14] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 392–401, 2014.

[KPB16] I. Karlsson, P. Papapetrou, and H. Bostrom. Generalized random shapelet forests. *Data Mining and Knowledge Discovery*, 30(5):1053–1085, Sep 2016.

[LDHB12] J. Lines, L. M. Davis, J. Hills, and A. Bagnall. A shapelet transform for time series classification. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 289–297, 2012.

[LL17] S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS)*, pages 4768–4777, 2017.

[LTB18] J. Lines, S. Taylor, and A. Bagnall. Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(5):52, 2018.

[RK13] T. Rakthanmanon and E. Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. pages 668–676, 05 2013.

[RSG16] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[RSG18] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1527–1535, 2018.

[SC78] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

[SS05] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005.

[Tav17] R. Tavenard. tslearn: A machine learning toolkit dedicated to time-series data, 2017. `https://github.com/rtavenar/tslearn`.

[THSD17] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 2962–2971, 2017.

[WYO17] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1578–1585, 2017.

[YK09] L. Ye and E. Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 947–956, 2009.

[ZKZ+18] J. Zhao, Y. Kim, K. Zhang, A. Rush, and Y. LeCun. Adversarially regularized autoencoders. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5902–5911, 2018.

Concomitant Lasso with Repetitions (CLaR): beyond averaging multiple realizations of heteroscedastic noise

Quentin Bertrand *[1], Mathurin Massias †[1], Alexandre Gramfort[1], and Joseph Salmon[2]

[1]INRIA, Université Paris-Saclay
[2]IMAG,Univ Montpellier, CNRS, Montpellier, France

May 31, 2019

Abstract

Sparsity promoting norms are frequently used in high dimensional regression. A limitation of Lasso-type estimators is that the regularization parameter depends on the noise level which varies between datasets and experiments. Estimators such as the concomitant Lasso address this dependence by jointly estimating the noise level and the regression coefficients. As sample sizes are often limited in high dimensional regimes, simplified heteroscedastic models are customary. However, in many experimental applications, data is obtained by averaging multiple measurements. This helps reducing the noise variance, yet it dramatically reduces sample sizes, preventing refined noise modeling. In this work, we propose an estimator that can cope with complex heteroscedastic noise structures by using non-averaged measurements and a concomitant formulation. The resulting optimization problem is convex, so thanks to smoothing theory, it is amenable to state-of-the-art proximal coordinate descent techniques that can leverage the expected sparsity of the solutions. Practical benefits are demonstrated on simulations and on neuroimaging applications.

Keys-words: inverse problems, sparsity, smoothing theory, neuroimaging.

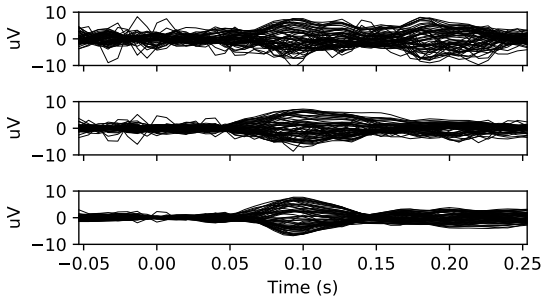


Figure 1: Amplitude $\bar{Y}$ of $n = 59$ EEG signals, averaged across $r = 5$ (top), $r = 10$ (middle), and $r = 50$ (bottom) repetitions. As the number of averaged repetitions increases, the noise is reduced and the measurements become smoother, revealing the brain response around 0.1 s.

1 Introduction

In many important statistical applications, the number of parameters p is much larger than the number of observations n. A popular approach to tackle linear regression problems in such high dimension scenarios is to consider convex ℓ_1-type penalties, as popularized by [Tib96]. The use of these penalties relies on a regularization parameter λ trading data fidelity versus sparsity. Unfortunately, statistical analysis reveals that the optimal λ should be proportional to the noise level ([BRT09]), which is rarely known in practice. To tackle this issue, one can jointly estimate the noise level and the regression coefficients. Such a concomitant estimation ([HD74, Hub81]) has recently been adapted for sparse regression by [Owe07] and analyzed under several names such as Square root Lasso ([BCW11]) or scaled Lasso ([SZ12]).

In these latter works, the noise parameter consists of a single variance parameter. However, in various applied settings, mixing data of different natures or coming from different sources is customary

*firstname.lastname@inria.fr
†firstname.lastname@inria.fr

to increase the number of observations. This often leads to heteroscedasticity: the data may be contaminated with non-uniform noise levels (differing across features or samples). This is the case for magneto-electroencephalographic (M/EEG) data, where observations come from three different types of sensors (gradiometers, magnetometers and electrodes), leading to very different amplitudes, noise levels and noise covariance matrices. Attempts to cope with heteroscedasticity were analyzed in this context by [DCL12, WD12, KS12, DHMS13]. Moreover, fast algorithms relying on smoothing techniques from the optimization community [Nes05] have been extended to heteroscedastic regression in a multi-task setting, through the Smooth Generalized Concomitant Lasso (SGCL, [MFGS18]). The SGCL is designed to jointly estimate the regression coefficients and the noise *co-standard deviation matrix*[1]. However, in certain applications, such as with M/EEG data, the number of parameters in the co-standard deviation matrix ($\approx 10^4$) is typically equal to the number of observations, making it statistically impossible to estimate accurately.

When observations are contaminated with a strong noise and the signal-to-noise ratio (SNR) is too low, provided measurements can be repeated, a natural idea is to average them. Indeed, under the assumption that the signal of interest is corrupted by some additive independant noise realizations, averaging different measurements divides the noise variance by the number of repetitions. This is classically done in experimental sciences from chemistry, to physics or neuroscience as it generally allows to visually inspect signals. This effect is illustrated in Figure 1 for electroencephalography (EEG) data. By averaging from 5 to 50 repetitions of the electrical response of the brain to a stimulus one can reveal a so-called evoked brain response around 100 ms after stimulation. It is usually this type of averaged data which is plugged into optimization solvers, hence discarding individual observations that could be used to better characterize the noise and improve the statistical estimation [GSH+13, OHG09].

In this work, we propose the Concomitant Lasso with Repetitions (CLaR), an estimator designed to exploit all available measurements collected during repetitions of experiments. The proposed concomitant formulation of the optimization problem derived has two strong benefits: first, the noise covariance is an explicit parameter of the model, on which it is easy to add structural constraints (*e.g.,* block diagonality) and second, smoothing theory leads to a cost function that can be minimized using efficient proximal coordinate descent techniques. By estimating the regression coefficients and the noise structure, this estimator demonstrates improvements in suppport identification compared to estimators using averaged data or assuming homoscedastic noise.

In Section 2, we recall the framework of concomitant estimation, and introduce our estimator. In Section 3, we detail the properties of CLaR, and derive an algorithm to solve it. Finally, Section 4 is dedicated to experimental results. All proofs can be found in the supplementary material: `https://arxiv.org/abs/1902.02509`.

[1] *i.e.,* the square root of the noise covariance matrix

2 Heteroscedastic concomitant estimation

Notation For a matrix $A \in \mathbb{R}^{m\times n}$ its jth column (*resp.* j^{th} row) is denoted $A_{:j} \in \mathbb{R}^{m\times 1}$ (*resp.* $A_{j:} \in \mathbb{R}^{1\times n}$). Let r be the number of repetitions of the experiment. The r observations matrices are denoted $Y^{(1)}, \dots, Y^{(r)} \in \mathbb{R}^{n\times q}$ with n being the number of sensors/samples and q corresponds, for example, to a number of tasks or a number of time samples. The mean over the repetitions of the observations matrices is written $\bar{Y} = \frac{1}{r}\sum_{l=1}^{r} Y^{(l)}$. Let $X \in \mathbb{R}^{n\times p}$ be the design matrix, with p features stored column-wise: $X = [X_{:1}, \dots, X_{:p}]$. The matrix $\mathrm{B} \in \mathbb{R}^{p\times q}$ contains the coefficients of the linear regression model. We write $\|\cdot\|$ (*resp.* $\langle\cdot,\cdot\rangle$) for the standard Euclidean norm (*resp.* inner product) on vectors and matrices, $\|\cdot\|_{p_1}$ for the ℓ_{p_1} norm, for any $p_1 \in [1,\infty)$. For a matrix $\mathrm{B} \in \mathbb{R}^{p\times q}$, $\|\mathrm{B}\|_{2,1} = \sum_{j=1}^{p} \|\mathrm{B}_{j:}\|$ (*resp.* $\|\mathrm{B}\|_{2,\infty} = \max_{j\in[p]} \|\mathrm{B}_{j:}\|$) is its row-wise norm, and for any $p_1 \in [0,\infty]$, we write $\|\mathrm{B}\|_{s,p_1}$ for the Schatten p_1-norm (*i.e.,* the ℓ_{p_1} norm of the singular values of B). The notation $\mathcal{S}_+^n$ (*resp.* $\mathcal{S}_{++}^n$) stands for the set of positive semi-definite matrices (*resp.* positive definite matrices). For S_1 and $S_2 \in \mathcal{S}_+^n$, $S_1 \succeq S_2$ if $S_1 - S_2 \in \mathcal{S}_+^n$, and $S_1 \succeq \underline{\sigma}$ if $S_1 \succeq \underline{\sigma}\,\mathrm{Id}_n$. For a square matrix $A \in \mathbb{R}^{n\times n}$, $\mathrm{Tr}(A)$ represents the trace of A and $\|A\|_S = \sqrt{\mathrm{Tr}(A^\top SA)}$ is the Mahalanobis norm induced by $S \in \mathcal{S}_{++}^n$. For $a, b \in \mathbb{R}$, we denote $(a)_+ = \max(a, 0)$, $a \vee b = \max(a, b)$ and $a \wedge b = \min(a, b)$. The block soft-thresholding operator at level $\tau > 0$, is denoted $\mathrm{BST}(\cdot, \tau)$, and reads for any vector x, $\mathrm{BST}(x, \tau) = (1 - \tau/\|x\|)_+ x$.

For $p_1 \in [1,\infty)$, let us write $\mathcal{B}_{p_1}$ for the associated unit ℓ_{p_1} ball. The identity matrix of size $n \times n$ is denoted Id_n, and $[r]$ is the set of integers from 1 to r.

2.1 Model and proposed estimator

We consider observations where r repetitions of the same experiment are performed, leading to measurements $Y^{(1)} \in \mathbb{R}^{n\times q}, \dots, Y^{(r)} \in \mathbb{R}^{n\times q}$. We assume each measurement follows the same linear model:

$$\forall l \in [r], \quad Y^{(l)} = X\mathrm{B}^* + S^*\mathrm{E}^{(l)} \ , \tag{1}$$

where the entries of $\mathrm{E}^{(l)}$ are *i.i.d.* standard normal distributions, the $\mathrm{E}^{(l)}$s are independent and $S^* \in \mathcal{S}^n_{++}$ is the *co-standard deviation matrix*, *i.e.,* the square root of the noise covariance matrix.[2] Note that even if the observations $Y^{(1)}, \dots, Y^{(r)}$ are different because of the noise $\mathrm{E}^{(1)}, \dots, \mathrm{E}^{(r)}$, the true parameter B^* and the noise structure S^* are shared across repetitions.

To leverage the multiple repetitions while taking into account the heteroscedasticity of the noise, we introduce the Concomitant Lasso with Repetitions estimator:

Definition 1 (Concomitant Lasso with Repetitions, CLaR)**.** CLaR estimates the parameters of (1) by solving

$$(\hat{\mathrm{B}}^{\mathrm{CLaR}}, \hat{\mathrm{S}}^{\mathrm{CLaR}}) \in \underset{\substack{\mathrm{B}\in\mathbb{R}^{p\times q}\\ S\succeq\underline{\sigma}}}{\arg\min}\, f(\mathrm{B}, S) + \lambda \|\mathrm{B}\|_{2,1} \ , \tag{2}$$

where $f(\mathrm{B}, S) := \frac{1}{2nqr}\sum_1^r \left\|Y^{(l)} - X\mathrm{B}\right\|^2_{S^{-1}} + \frac{1}{2n}\operatorname{Tr}(S)$, $\lambda > 0$ controls the sparsity of $\hat{\mathrm{B}}^{\mathrm{CLaR}}$ and $\underline{\sigma} > 0$ controls the smallest eigenvalue of $\hat{\mathrm{S}}^{\mathrm{CLaR}}$.

2.2 Connections with previous estimator

In low SNR settings, the standard way to deal with strong noise is to use the averaged observation $\bar{Y} \in \mathbb{R}^{n\times q}$ instead of the raw observations. The associated model reads:

$$\bar{Y} = X\mathrm{B}^* + \tilde{S}^*\tilde{\mathrm{E}} \ , \tag{3}$$

with $\tilde{S}^* := \frac{1}{\sqrt{r}}S^*$ and $\tilde{\mathrm{E}}$ has *i.i.d.* entries drawn from a standard normal distribution. The SNR of the average is multiplied by $\sqrt{r}$, yet the number of available samples to characterize the noise goes from rq to q. This explains why averaging is not sufficient to estimate complex heteroscedastic noise models, *i.e.,* when S^* is far from a scalar matrix. Our introduced CLaR is a generalization of the Smoothed Generalized Concomitant Lasso ([MFGS18]) which only targets averaged observations:

[2] since we impose $S^* \in \mathcal{S}^n_{++}$, it is uniquely defined

Definition 2 (Smoothed Generalized Concomitant Lasso, SGCL)**.** SGCL estimates the parameters of model (3), by solving:

$$(\hat{\mathrm{B}}^{\mathrm{SGCL}}, \hat{\mathrm{S}}^{\mathrm{SGCL}}) \in \underset{\substack{\mathrm{B}\in\mathbb{R}^{p\times q}\\ \tilde{S}\succeq\underline{\sigma}/\sqrt{r}}}{\arg\min}\, \tilde{f}(\mathrm{B}, \tilde{S}) + \lambda \|\mathrm{B}\|_{2,1} \ , \tag{4}$$

with $\tilde{f}(\mathrm{B}, \tilde{S}) := \frac{1}{2nq}\|\bar{Y} - X\mathrm{B}\|^2_{\tilde{S}^{-1}} + \frac{1}{2n}\operatorname{Tr}(\tilde{S})$.

Remark 1. Note that $\hat{\mathrm{S}}^{\mathrm{CLaR}}$ estimates S^*, while $\hat{\mathrm{S}}^{\mathrm{SGCL}}$ estimates $\tilde{S}^* = S^*/\sqrt{r}$. Since we impose the constraint $\hat{\mathrm{S}}^{\mathrm{CLaR}} \succeq \underline{\sigma}$, we adapt the scaling so that $\hat{\mathrm{S}}^{\mathrm{SGCL}} \succeq \underline{\sigma}/\sqrt{r}$ in (4) for future comparisons.

Remark 2. SGCL is a particular case of CLaR: for $r = 1$ and $Y^{(1)} = \bar{Y}$, the solution of CLaR is the same as the one of SGCL.

The justification for the introduction of CLaR is the following: using the quadratic loss $\|Y - X\mathrm{B}\|^2$, the parameters of model (1) can be estimated by using either $\|\bar{Y} - X\mathrm{B}\|^2$ or $\frac{1}{r}\sum\|Y^{(l)} - X\mathrm{B}\|^2$ as a data-fitting term. It turns out that in such a case, the two alternatives yield the same solutions (as the two terms are equal up to constants in B). Hence, apart from averaging, the quadratic loss does not leverage the multiple repetitions, and ignores the noise structure. Using the data-fitting term of CLaR allows to incorporate this structure.

2.3 Smoothing of the nuclear norm

In this section we shed some light on the properties of CLaR, especially with respect to smoothing theory ([Nes05, BT12]). The following proposition relates the data-fitting term used in CLaR and SGCL showing the link with the smoothing of the Schatten 1-norm (*a.k.a.* the trace norm or the nuclear norm). For that, we introduce the following smoothing function:

$$\omega_{\underline{\sigma}}(\cdot) = \frac{\underline{\sigma}}{2}\|\cdot\|^2_F + \underline{\sigma}\frac{n\wedge q}{2}, \tag{5}$$

and the inf-convolution of functions f and g, defined as $f \,\square\, g(y) = \inf_x f(x) + g(y - x)$.

Proposition 1. The $\omega_{\underline{\sigma}}$-smoothing of the Schatten-1 norm, *i.e.,* the function $\|\cdot\|_{s,1} \,\square\, \omega_{\underline{\sigma}} : \mathbb{R}^{n\times q} \mapsto \mathbb{R}$, is the solution of the following (smooth) optimization problem:

$$(\|\cdot\|_{s,1} \,\square\, \omega_{\underline{\sigma}})(Z) = \min_{S\succeq\underline{\sigma}} \tfrac{1}{2}\|Z\|^2_{S^{-1}} + \tfrac{1}{2}\operatorname{Tr}(S) \ . \tag{6}$$

Proof of Proposition 1 is in Supplementary A.3.

Definition 3 (Clipped Square Root)**.** For $S \in \mathcal{S}_+^n$, let us define the *Clipped Square Root* operator:

$$\mathrm{ClSqrt}(S, \underline{\sigma}) = U \operatorname{diag}(\sqrt{\gamma_1} \vee \underline{\sigma}, \dots, \sqrt{\gamma_n} \vee \underline{\sigma}) U^\top \ , \quad (7)$$

where $S = U \operatorname{diag}(\gamma_1, \dots, \gamma_n) U^\top$ and U is orthogonal.

Remark 3. Note that $\mathrm{ClSqrt}(S, \underline{\sigma})$ is the projection of the square root of S onto the affine cone $\{S \in \mathcal{S}_{++}^n : S \succeq \underline{\sigma}\}$.

We can now state explicitly the connection between the SGCL, CLaR and the Schatten 1-norm.

Proposition 2 (Smoothing properties of CLaR)**.** The solution of the CLaR problem in Equation (2), $(\hat{\mathrm{B}}, \hat{S}) = (\hat{\mathrm{B}}^{\mathrm{CLaR}}, \hat{S}^{\mathrm{CLaR}})$ is a solution of:

$$\hat{\mathrm{B}} = \underset{\mathrm{B} \in \mathbb{R}^{p \times q}}{\arg\min} (\|\cdot\|_{s,1} \,\square\, \omega_{\underline{\sigma}})(Z) + \lambda n \|\mathrm{B}\|_{2,1}$$
$$\hat{S} = \mathrm{ClSqrt}(\tfrac{1}{r} Z Z^\top, \underline{\sigma}) \ ,$$

where $Z = [Z^{(1)} | \dots | Z^{(r)}]$ and $Z^{(l)} = \frac{Y^{(l)} - X\mathrm{B}}{\sqrt{q}}$.

Proof of Proposition 2 can be found in Appendix B.1.

Remark 4. Following Remark 1, similar properties can be obtained for $\hat{\mathrm{B}}^{\mathrm{SGCL}}$ and $\hat{S}^{\mathrm{SGCL}}$ letting $r = 1$, and substituting $\bar{Z} := \frac{\bar{Y} - X\mathrm{B}}{\sqrt{q}}$ to Z in the former proposition.

Ideas similar to the ones of Propositions 1 and 2 can be traced to [van16, Lemma 3.4, p. 37], where the following formulation was introduced to prove oracle inequalities for the multivariate square-root Lasso[3]:

$$\|Z\|_{s,1} = \min_{S \in \mathcal{S}_{++}^n} \frac{1}{2} \|Z\|_{S^{-1}}^2 + \frac{1}{2} \operatorname{Tr}(S) \ . \quad (8)$$

In the present contribution, the problem formulation in Proposition 1 is motivated by computational aspects, as it helps to address the combined non-differentiability of the data-fitting term $\|\cdot\|_{s,1}$ and the penalty $\|\cdot\|_{2,1}$ term.

Other alternatives to exploit the multiple repetitions without simply averaging them, would consist in investigating other Schatten p_1-norms:

$$\arg\min_{\mathrm{B}} \frac{1}{\sqrt{rq}} \|[Y^{(1)} - X\mathrm{B} | \dots | Y^{(r)} - X\mathrm{B}]\|_{s,p_1} + \lambda n \|\mathrm{B}\|_{2,1} \quad (9)$$

Without smoothing, problems of the form given in Equation (9) have the drawback of having no smooth term, and solvers have to resort to proximal splitting algorithms (*e.g.,* the ones by [DR56] or [CP11]) that can handle the sum of two non-smooth components.

[3] defined as the solution of Equation (9) with $p_1 = 1$

Even if the non-smooth Schatten 1-norm is replaced by the formula in (8), numerical challenges remain: S can approach 0 arbitrarily, hence, the gradient *w.r.t.* S of the data-fitting term is not Lipschitz over the optimization domain. A similar problem was raised by [NFG+17] for the concomitant Lasso and leads to the introduction of smoothing techniques to address it.

Here we replaced the Schatten norm with $p_1 = 1$ by its smoothed version $\|\cdot\|_{s,p_1} \,\square\, \omega_{\underline{\sigma}}$, for some smooth function $\omega_{\underline{\sigma}}$. Results for other Schatten p_1-norms are provided in the Appendix; see Proposition 11 (*resp.* Proposition 12) for the case of the Schatten 2-norm (*resp.* Schatten ∞-norm).

3 Properties of CLaR

We detail the principal results needed to solve Problem (2) numerically in this section, leading to the efficient implementation proposed in Algorithm 1. First we show that CLaR is based on a convex and smooth optimization problem that can be efficiently solved through Block Coordinate Descent. Then we prove some statistical properties on CLaR, in particular that the estimation of the covariance matrix is improved by a factor r in CLaR.

3.1 Alternate minimization

Proposition 3. CLaR is jointly convex in B and S so minimizing the objective alternatively in S and in B (see Algorithm 1) converges to a global minimum.

Proof.

$$f(\mathrm{B}, S) = \frac{1}{2nqr} \sum_1^r \left\| Y^{(l)} - X\mathrm{B} \right\|_{S^{-1}}^2 + \frac{1}{2n} \operatorname{Tr}(S)$$
$$= \operatorname{Tr}(Z^T S^{-1} Z) + \frac{1}{2n} \operatorname{Tr}(S) \ ,$$

with $Z = \frac{1}{\sqrt{2nqr}} [Y^{(1)} - X\mathrm{B} | \dots | Y^{(r)} - X\mathrm{B}]$.

$(Z, \Sigma) \mapsto \operatorname{Tr} Z^\top \Sigma^{-1} Z$ is jointly convex over $\mathbb{R}^{n \times q} \times \mathbb{S}_{++}^n$, see [BV04, Example 3.4]. This means that f is jointly convex in (Z, S), moreover $\mathrm{B} \mapsto \frac{1}{\sqrt{2nqr}} [Y^{(1)} - X\mathrm{B} | \dots | Y^{(r)} - X\mathrm{B}]$ is linear in B, thus f is jointly convex in (B, S), meaning that $f + \lambda g$ is jointly convex in (B, S). Moreover the constraint set is convex and thus solving CLaR is a convex problem. $\square$

As the next proposition shows, optimizing over S with B being fixed involves the Clipped Square Root operator of Definition 3.

Proposition 4 (Minimization in S)**.** Let $B \in \mathbb{R}^{n\times q}$ be fixed. The minimization of $f(\mathrm{B}, S)$ *w.r.t.* S with the constraint $S \succeq \underline{\sigma}$ admits the closed-form solution:

$$S = \mathrm{ClSqrt}\left(\frac{1}{r}\sum_{l=1}^{r} Z^{(l)}Z^{(l)\top}, \underline{\sigma}\right) \quad , \tag{10}$$

with $Z^{(l)} = \frac{1}{\sqrt{q}}(Y^{(l)} - X\mathrm{B})$.

Proof. Minimizing $f(\mathrm{B}, S)$ in S amounts to solving

$$\underset{S\succeq\underline{\sigma}}{\arg\min}\ \tfrac{1}{2}\left\|Z\right\|_{S^{-1}}^2 + \tfrac{1}{2}\mathrm{Tr}(S) \quad . \tag{11}$$

with $Z = \frac{1}{\sqrt{r}}[Z^{(1)}|\dots|Z^{(l)}]$. The solution is $\mathrm{ClSqrt}\left(ZZ^\top, \underline{\sigma}\right)$ (see [MFGS18, Appendix A2]), and $ZZ^\top = \frac{1}{r}\sum_{l=1}^{r} Z^{(l)}Z^{(l)\top}$. □

We can now state the update of the B term in the alternate minimization process. Here, because of the row-wise separability of $\|\cdot\|_{2,1}$, we use block updates over rows, whose closed-form are provided by the next proposition:

Proposition 5. Each step of the block minimization of $f(\cdot, S) + \lambda\left\|\cdot\right\|_{2,1}$ in the jth line of B admits a closed-form solution:

$$\mathrm{B}_{j:} = \mathrm{BST}\left(\mathrm{B}_{j:} + \frac{X_{:j}^\top S^{-1}(Y - X\mathrm{B})}{\left\|X_{:j}\right\|_{S^{-1}}^2}, \frac{\lambda nq}{\left\|X_{:j}\right\|_{S^{-1}}^2}\right) \quad . \tag{12}$$

Proof. The function to minimize is the sum of a smooth term $f(\cdot, S)$ and a non-smooth but separable term, $\|\cdot\|_{2,1}$, whose proximal operator can be computed:

- f is $\left\|X_{:j}\right\|_{S^{-1}}^2/nq$-smooth with respect to $\mathrm{B}_{j:}$, with partial gradient $\nabla_j f(\cdot, S) = -\frac{1}{nq}X_{:j}^\top S^{-1}(\bar{Y} - X\mathrm{B})$,
- $\|\mathrm{B}\|_{2,1} = \sum_{j=1}^{p}\|\mathrm{B}_{j:}\|$ is row-wise separable over B, with $\mathrm{prox}_{\lambda nq/\|X_{:j}\|_{S^{-1}}^2, \|\cdot\|}(\cdot) = \mathrm{BST}\left(\cdot, \frac{\lambda nq}{\|X_{:j}\|_{S^{-1}}^2}\right)$.

Hence, proximal block-coordinate descent converges ([TY09]), and the update reads like Equation (12). The closed-form formula arises since the smooth part of the objective is quadratic and isotropic *w.r.t.* $\mathrm{B}_{j:}$. □

3.2 Critical parameter, duality gap and stopping criterion

As for the Lasso we show that there is a critical parameter, *i.e.,* there exists $\lambda_{\max} \geq 0$ such that whenever λ is greater than this value, the coefficients recovered vanish:

Proposition 6 (Critical regularization parameter)**.** For the CLaR estimator we have the following property: with $S_{\max} = \mathrm{ClSqrt}(\frac{1}{qr}\sum_{l=1}^{r} Y^{(l)}Y^{(l)\top}, \underline{\sigma})$,

$$\hat{\mathrm{B}} = 0,\ \ \forall\lambda \geq \lambda_{\max} := \frac{1}{nq}\left\|X^\top S_{\max}^{-1}\bar{Y}\right\|_{2,\infty} \quad . \tag{13}$$

Proof. First notice that if $\hat{\mathrm{B}} = 0$, then $\hat{S} = \mathrm{ClSqrt}(\frac{1}{qr}\sum_{l=1}^{r} Y^{(l)}Y^{(l)\top}, \underline{\sigma}) := S_{\max}$.

Fermat's rules states that

$$\begin{aligned}\hat{\mathrm{B}} = 0 &\Leftrightarrow 0 \in \partial\left(f(\cdot, S_{\max}) + \lambda\|\cdot\|_{2,1}(0\right)\\ &\Leftrightarrow -\nabla f(\cdot, S_{\max}) \in \lambda\mathcal{B}_{\|\cdot\|_{2,\infty}}\\ &\Leftrightarrow \frac{1}{nq}\left\|X^\top S_{\max}^{-1}\bar{Y}\right\|_{2,\infty} := \lambda_{\max} \leq \lambda \quad .\end{aligned} \tag{14}$$

□

To ensure the convergence of our algorithm, we use the duality gap as a stopping criterion (as it guarantees a targeted sub-optimality level). For its computation we therefore need to explicitely state the dual optimization problem of Problem (2):

Proposition 7. With $\hat{\Theta} = (\hat{\Theta}^{(1)}, \dots, \hat{\Theta}^{(r)})$, the dual formulation of Problem (2) is

$$\begin{aligned}\hat{\Theta} = \underset{(\Theta^{(1)},\dots,\Theta^{(r)})\in\Delta_{X,\lambda}}{\arg\max}\ &\frac{\underline{\sigma}}{2}\left(1 - \frac{qn\lambda^2}{r}\sum_{l=1}^{r}\mathrm{Tr}\,\Theta^{(l)}\Theta^{(l)\top}\right)\\ &+ \frac{\lambda}{r}\sum_{l=1}^{r}\left\langle\Theta^{(l)}, Y^{(l)}\right\rangle \quad ,\end{aligned} \tag{15}$$

with $\bar{\Theta} = \frac{1}{r}\sum_1^r \Theta^{(l)}$ and

$$\begin{aligned}\Delta_{X,\lambda} = \Big\{&(\Theta^{(1)}, \dots, \Theta^{(r)}) \in (\mathbb{R}^{n\times q})^r :\\ &\left\|X^\top\bar{\Theta}\right\|_{2,\infty} \leq 1, \left\|\sum_{l=1}^{r}\Theta^{(l)}\Theta^{(l)\top}\right\|_2 \leq \frac{r}{\lambda^2 n^2 q}\Big\} \quad .\end{aligned} \tag{16}$$

Proof of Proposition 7 is in Supplementary B.2.

In Algorithm 1 the dual point Θ at iteration t is obtained through the relations $\Theta^{(l)} = \frac{1}{nq\lambda}(Y^{(l)} - X\mathrm{B})$ (with B the current primal iterate), and then projected on $\Delta_{X,\lambda}$.

Remark 5. Once the quantity $\mathrm{cov}_Y := \frac{1}{r}\sum_1^r Y^{(l)}Y^{(l)\top}$ has been pre-computed, the cost of updating S in CLaR does not depend on r, hence is the same as working with averaged data. Indeed, R being defined as $R = [Y^{(1)} - X\mathrm{B}|\dots|Y^{(r)} - X\mathrm{B}]$, computing $RR^\top = \mathrm{RRT}(\mathrm{cov}_Y, Y, X, \mathrm{B}) := r\mathrm{cov}_Y + r(X\mathrm{B})(X\mathrm{B})^\top - r\bar{Y}^\top(X\mathrm{B}) - r(X\mathrm{B})^\top\bar{Y}$ can be done in $\mathcal{O}(qn^2)$ (details are in Supplementary B.3).

Algorithm 1 ALTERNATE MIN. FOR CLAR

input : $X, \bar{Y}, \underline{\sigma}, \lambda, f, T$
init : $\mathrm{B} = 0_{p,q}$, $S^{-1} = \underline{\sigma}^{-1}\,\mathrm{Id}_n$, $\bar{R} = \bar{Y}$
$\mathrm{cov}_Y = \frac{1}{r}\sum_{l=1}^{r} Y^{(l)}Y^{(l)\top}$ `// precomputed`
for iter $= 1, \dots, T$ **do**
  if iter $= 1 \pmod f$ **then** `// noise update`
    $RR^\top = \mathrm{RRT}(\mathrm{cov}_Y, Y, X, \mathrm{B})$ `// Rk. (5)`
    $S \leftarrow \mathrm{ClSqrt}(\frac{1}{qr}RR^\top, \underline{\sigma})$ `// Eq. (10)`
    for $j = 1, \dots, p$ **do** $L_j = X_{:j}^\top S^{-1} X_{:j}$
  for $j = 1, \dots, p$ **do** `// coef. update`
    $\bar{R} \leftarrow \bar{R} + X_{:j}\mathrm{B}_{j:}$
    $\mathrm{B}_{j:} \leftarrow \mathrm{BST}\Big(\dfrac{X_{:j}^\top S^{-1}\bar{R}}{L_j}, \dfrac{\lambda nq}{L_j}\Big)$
    $\bar{R} \leftarrow \bar{R} - X_{:j}\mathrm{B}_{j:}$
return B, S

3.3 Statistical comparison

In this section, we show the statistical interest of using all repetitions of the experiments instead of using a mere averaging as SGCL would do (remind that the later is equivalent to CLaR with $r = 1$ and $Y^{(1)} = \bar{Y}$, see Remark 2).

Let us introduce Σ^*, the true covariance matrix of the noise (*i.e.,* $\Sigma^* = S^{*2}$ with our notation). In SGCL and CLaR alternate minimization consists in a succession of estimations of B^* and Σ^* (more precisely $S = \mathrm{ClSqrt}(\Sigma, \underline{\sigma})$ is estimated along the process). In this section we explain why the estimation of Σ^* provided by CLaR has some more interesting statistical properties with respect to one obtained by SGCL. For that, we can compare the estimates of Σ^* one would obtain provided that the true parameter B^* is known by both SGCL and CLaR. In such "ideal" scenario, the associated estimators of Σ^* could be written:

$$\hat{\Sigma}^{\mathrm{CLaR}} := \frac{1}{qr}\sum_{l=1}^{r}(Y^{(l)} - X\hat{\mathrm{B}})(Y^{(l)} - X\hat{\mathrm{B}})^\top \ , \tag{17}$$

$$\hat{\Sigma}^{\mathrm{SGCL}} := \frac{1}{qr}\Big(\sum_{l=1}^{r} Y^{(l)} - X\hat{\mathrm{B}}\Big)\Big(\sum_{l=1}^{r} Y^{(l)} - X\hat{\mathrm{B}}\Big)^\top , \tag{18}$$

with $\hat{\mathrm{B}} = \mathrm{B}^*$, and satisfy the following properties:

Proposition 8. Provided that the true signal is known, and that the covariance estimator $\hat{\Sigma}^{\mathrm{CLaR}}$ and $\hat{\Sigma}^{\mathrm{SGCL}}$ are defined thanks to Equations (17) and (18), then one can check that

$$\mathbb{E}(\hat{\Sigma}^{\mathrm{CLaR}}) = \mathbb{E}(\hat{\Sigma}^{\mathrm{SGCL}}) = \Sigma^* \ , \tag{19}$$

$$\mathrm{cov}(\hat{\Sigma}^{\mathrm{CLaR}}) = \frac{1}{r}\mathrm{cov}(\hat{\Sigma}^{\mathrm{SGCL}}) \ . \tag{20}$$

Table 1: Algorithms cost

	CD epoch	gap computation
CLaR	$\mathcal{O}(\frac{n^3+qn^2}{f} + pn^2 + pnq)$	$\mathcal{O}(rnq + p)$
SGCL	$\mathcal{O}(\frac{n^3+qn^2}{f} + pn^2 + pnq)$	$\mathcal{O}(nq + p)$
MTLR	$\mathcal{O}(npqr)$	$\mathcal{O}(rnq + p)$
MTL	$\mathcal{O}(npq)$	$\mathcal{O}(nq + p)$

Proof of Proposition 8 can be found in Supplementary B.4

Proposition 8 states that $\hat{\Sigma}^{\mathrm{CLaR}}$ and $\hat{\Sigma}^{\mathrm{SGCL}}$ are unbiased estimators of Σ^* but our newly introduced CLaR, improves the estimation of the covariance structure by a factor r, the number of repetitions performed.

Empirically[4], we have also observed that $\hat{\Sigma}^{\mathrm{CLaR}}$ has larger eigenvalues than $\hat{\Sigma}^{\mathrm{SGCL}}$, leading to a less biased estimation of S^* after clipping the singular values.

4 Experiments

The code is released as an open source package and can be found here `https://github.com/QB3/CLaR`. The implementation is in Python, with Numba compilation ([LPS15]) to increase the speed of the algorithm.

Comparison with other estimators We compare CLaR to other estimators: SGCL, the Multi-Task Lasso (MTL, [OTJ10]) and a version of the MTL with repetitions: MTLR. Recall that CLaR solves Problem (2) and that SGCL solves Problem (4); we also remind the definition of MTL and MTLR:

$$\hat{\mathrm{B}}^{\mathrm{MTL}} \in \underset{\mathrm{B}\in\mathbb{R}^{p\times q}}{\arg\min} \frac{1}{2nq}\left\|\bar{Y} - X\mathrm{B}\right\|^2 + \lambda\left\|\mathrm{B}\right\|_{2,1} \ . \tag{21}$$

With $Y = [Y^{(1)}|\dots|Y^{(r)}] \in \mathbb{R}^{n\times rq}$, MTLR first solves:

$$\hat{\mathrm{B}}^{C} \in \underset{\mathrm{B}\in\mathbb{R}^{p\times rq}}{\arg\min} \frac{1}{2nq}\|Y - X\mathrm{B}\|^2 + \lambda\left\|\mathrm{B}\right\|_{2,1} \ , \tag{22}$$

then the estimator is defined as $\hat{\mathrm{B}}^{\mathrm{MTLR}} = \frac{1}{r}\sum_{l=1}^{r}\hat{\mathrm{B}}^{(l)}$, with $\hat{\mathrm{B}}^{C} = [\hat{\mathrm{B}}^{(1)}|\dots|\hat{\mathrm{B}}^{(r)}]$.

The cost of an epoch of coordinate descent and the cost of computing the gap for each for each algorithm are summarized in Table 1.

[4] in that case we plug $\hat{\mathrm{B}} = \hat{\mathrm{B}}^{\mathrm{CLaR}}$ (resp. $\hat{\mathrm{B}} = \hat{\mathrm{B}}^{\mathrm{CLaR}}$) in Proposition 8

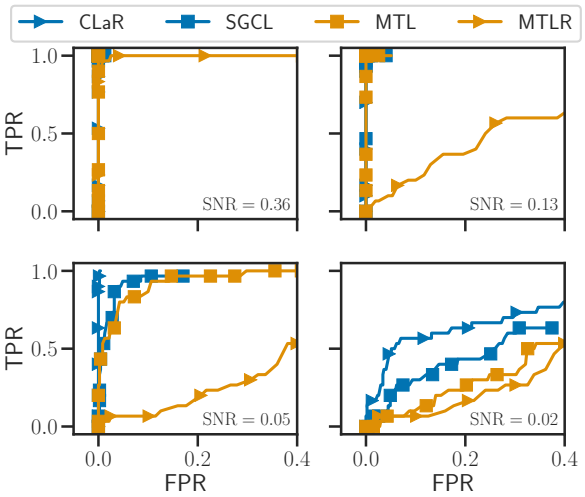


Figure 2: ROC curves of true support recovery for the CLaR, SGCL, MTL and MTLR with $\rho_X = 0.6$, $\rho_S = 0.6$, $r = 50$ for different values of SNR.

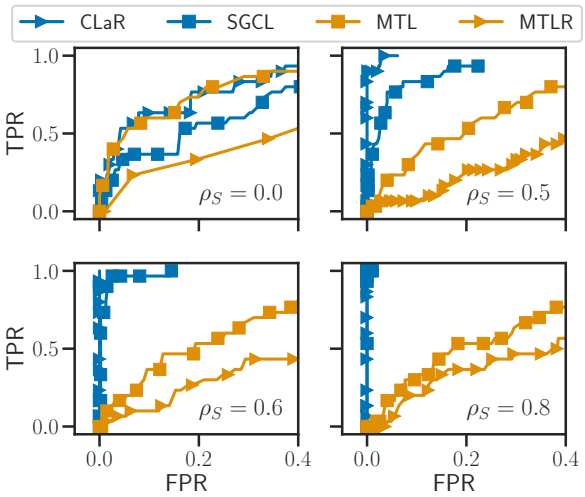


Figure 3: ROC curves of true support recovery for the CLaR, SGCL, MTL and MTLR with $\rho_X = 0.6$, SNR $= 0.03$, $r = 50$ for different values of ρ_S.

4.1 Support recovery

Here we demonstrate the ability of our estimator to recover the support *i.e.,* the ability to identify the predictive features. There are $n = 150$ observations, $p = 500$ features, $q = 100$ tasks. The design X is random with Toeplitz-correlated features with parameter $\rho_X = 0.6$ (correlation between $X_{:i}$ and $X_{:j}$ is $\rho_X^{|i-j|}$), and its columns have unit Euclidean norm. The true coefficient B^* has 30 non-zeros rows whose entries are independent and normally centered distributed. S^* is a Toeplitz matrix with parameter ρ_S. The SNR is fixed and constant across all repetitions $\mathrm{SNR} := \|X\mathrm{B}^*\|/\|X\mathrm{B}^* - Y^{(l)}\|$.

For Figures 2 to 4, fhe figure of merit is the ROC curve, *i.e.,* the true positive rate against the false positive rate. For the four estimators, the ROC curve is obtained by varying the value of the regularization parameter λ on a geometric grid of 160 points, starting from $\lambda_{\max}$ (specific to each algorithm) to $\lambda_{\min}$, the latest being also specific and chosen to obtain large enough false positive rates.

SNR influence On Figure 2 we can see that when the SNR is high (top left), all curves reach the (0, 1) point. This means that for each algorithm, there exists a λ such that the estimated support is exactly the true one. However, when the SNR decreases (top right, bottom left), the performance of SGCL, MTL and MTLR starts to drop, while that of CLaR remains stable. Finally, when the SNR is too low (bottom right), all algorithms perform poorly, but CLaR still performs better. This highlights the capacity of CLaR to leverage multiple repetitions of measurements to finely estimate the heteroscedastic noise.

Noise structure influence Figure 3 represents the ROC curves of CLaR, SGCL, MTL and MTLR for different values of ρ_S. As ρ_S increases, the noise becomes less and less heteroscedastic: from top left to bottom right, the performance of CLaR and SGCL increases as they are designed to exploit correlations in the noise, while the performance of MTL and MTLR decreases, as their homoscedastic model becomes less and less valid.

Influence of the number of repetitions Figure 4 shows ROC curves of CLaR, SGCL, MTL and MTLR for different values of r, starting from $r = 1$, where CLaR and SGCL are equivalent (as well as MTL and MTLR), to $r = 100$. It can be seen that even with $r = 20$ CLaR outperforms the other estimators, and that with $r = 100$ CLaR benefits more than any of the other algorithms from the large number of repetitions.

4.2 Real data

We now evaluate our estimators on real magneto and electroencephalography (M/EEG) data. The M/EEG

recordings measure the electrical potential and magnetic field induced by the active neurons. Data are time series of length q with n sensors and p sources (locations in the brain). Because the propagation of the electromagnetic fields is driven by the linear Maxwell equations, one can assume that the relation between the measurements $Y^{(1)}, \dots, Y^{(r)}$ and the amplitudes of sources in the brain B^* is linear. The M/EEG inverse problem consists in identifying B^*. Because of the limited number of sensors (a few hundred in practice), as well as the physics of the problem, the M/EEG inverse problem is severely ill-posed and needs regularization that provides plausible biological solutions. Because the experiments are usually short (less than 1 s.) and focused on specific cogntive functions, the number of active sources is expected to be small, *i.e.,* B^* is assumed to be row-sparse. Thus the M/EEG inverse problem fits the framework of Section 2.

We use the *sample* dataset from MNE ([GLL+14]). The experimental conditions are here auditory stimulations in the right or left ears or visual stimulations in the right or left visual fields, leading to different active locations in the brain (*i.e.,* different B^* for each type of stimulation). For this experiment, we keep only the gradiometer magnetic channels, and we reject one of them due to strong artifacts. This leads to 203 gradiometers, *i.e.,* $n = 203$ signals. The length of the temporal series is $q = 30$, and the data contains $r = 50$ repetitions. We choose a source space of size $p = 1281$ (oct-3 resolution). The orientation is fixed, and normal to the cortical mantle.

To generate the semi-real data we use the real design matrix X and the real co-standard deviation matrix S, estimated on pre-stimulation data. We then generate plausible MEG signals with MNE. The signals being contaminated with correlated noise, if one wants to use homoscedastic solvers it is necessary to whiten the data first (and thus to have an estimation of the covariance matrix, the later often being poor or not known). In this experiment we demonstrate that without this whitening process, the homoscedastic solver MTL fails, whereas CLaR succeeds (we dropped here MTLR since it performed poorly on synthetic data and was way to slow to converge).

However, it would not make sense to apply our solver directly on the design matrix X. Here we describe the preprocessing applied to X and Y, using information from X only. First we pre-whiten the data Y and the design matrix X by multiplying from the left by the whitener matrix $W_0 = \mathrm{diag}((\|X_{i:}\|)_{i\in[n]})$. Then, we rescale each column of the design matrix X to have a (Euclidean) norm of 1. Finally, as in Section 4.1, Figure 5 is obtained by varying the estimator-specific regularization parameter λ from $\lambda_{\max}$ to $\lambda_{\min}$ on a geometric grid.

Results of this experiment are in Figure 5. With (top) 2 active sources in the brain, 1 auditory left and 1 auditory right (*i.e.,* $\|\mathrm{B}\|_{2,1}^* = 2$), the MTL estimator performs poorly and does not recover the full support before reaching a false positive rate (FPR) of 0.18. It makes sense since the MTL is not designed to cope with heteroscedastic noise, and the data is not whitened in this experiment. SGCL also performs poorly, because of its statistical limitations: the number of observations used to estimate the covariance matrix in SGCL is too low (here $n \times q \approx 6 \times 10^3$ observations to estimate $n^2/2 \approx 2\times10^4$ parameters). CLaR performs better and is able to recover the true sources with a lower FPR. It is worth noting that in CLaR $r \times q \times n \approx 3 \times 10^5$ observations are used to estimate the $n^2/2 \approx 2 \times 10^4$ parameters of the covariance matrix.

Figure 5 (bottom) shows an experiment with 3 active sources, 1 auditory left, 1 auditory right and 1 visual left, CLaR can recover 2 sources among 3 with a low FPR, whereas SGCL and CLaR do not identify 2/3 the sources before at least a FPR of 0.15.

Conclusion This work introduces CLaR, a sparse estimator for multitask regression. It is designed to handle heteroscedastic noise in the context of repeated observations. The resulting optimization problem can be solved efficiently with state-of-the-art convex solvers. The theory of smoothing connects CLaR to the Schatten 1-Lasso in a principled manner, which opens the way to the use of more sophisticated datafitting terms.

Acknowledgments This work was funded by ERC Starting Grant SLAB ERC-YStG-676943.

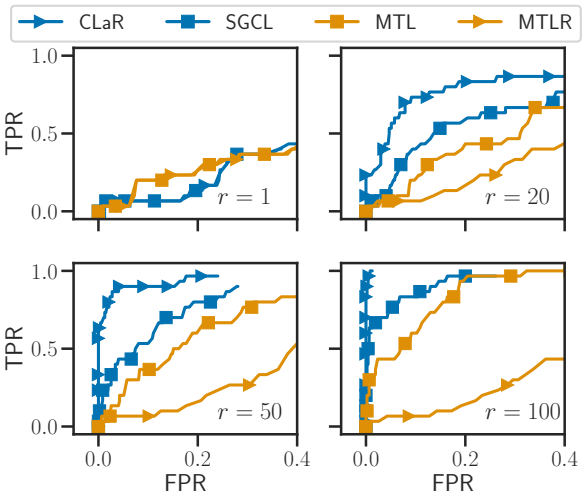


Figure 4: ROC curves of true support recovery for the CLaR, SGCL, MTL and MTLR with $\rho_X = 0.6$, $\rho_S = 0.4$, SNR $= 0.03$, for different values of r.

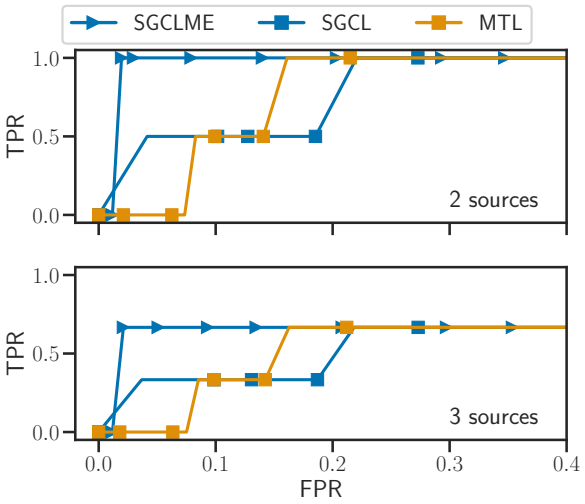


Figure 5: ROC curves of true support recovery for (top) 2 sources: one left auditory and one right auditory source, and for (bottom) 3 sources : one left auditory, one right auditory and one left visual source for the CLaR, SGCL and MTL with real M/EEG design and noise.

References

[BCW11] A. Belloni, V. Chernozhukov, and L. Wang. Square-root Lasso: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98(4):791–806, 2011.

[Bec17] A. Beck. *First-Order Methods in Optimization*, volume 25. SIAM, 2017.

[BRT09] P. J. Bickel, Y. Ritov, and A. B. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Ann. Statist.*, 37(4):1705–1732, 2009.

[BT12] A. Beck and M. Teboulle. Smoothing and first order methods: A unified framework. *SIAM J. Optim.*, 22(2):557–580, 2012.

[BV04] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[CP11] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vis.*, 40(1):120–145, 2011.

[DCL12] J. Daye, J. Chen, and H. Li. High-dimensional heteroscedastic regression with an application to eQTL data analysis. *Biometrics*, 68(1):316–326, 2012.

[DHMS13] A. S. Dalalyan, M. Hebiri, K. Meziani, and J. Salmon. Learning heteroscedastic models by convex programming under group sparsity. In *ICML*, 2013.

[DR56] J. Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 82(2):421–439, 1956.

[GLL+14] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, and M. S. Hämäläinen. MNE software for processing MEG and EEG data. *NeuroImage*, 86:446–460, 2014.

[GSH+13] A. Gramfort, D. Strohmeier, J. Haueisen, M. S. Hämäläinen, and M. Kowalski. Time-frequency mixed-norm estimates: Sparse M/EEG imaging with non-stationary source activations. *NeuroImage*, 70:410–422, 2013.

[HD74] P. J. Huber and R. Dutter. Numerical solution of robust regression problems. In *Compstat 1974 (Proc. Sympos. Computational Statist., Univ. Vienna, Vienna,*

1974), pages 165–172. Physica Verlag, Vienna, 1974.

[Hub81] P. J. Huber. *Robust Statistics*. John Wiley & Sons Inc., 1981.

[KS12] M. Kolar and J. Sharpnack. Variance function estimation in high-dimensions. In *ICML*, pages 1447–1454, 2012.

[LPS15] S. K. Lam, A. Pitrou, and S. Seibert. Numba: A LLVM-based Python JIT Compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6. ACM, 2015.

[MFGS18] M. Massias, O. Fercoq, A. Gramfort, and J. Salmon. Generalized concomitant multitask lasso for sparse multimodal regression. In *AISTATS*, volume 84, pages 998–1007, 2018.

[Nes05] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, 2005.

[NFG+17] E. Ndiaye, O. Fercoq, A. Gramfort, V. Leclère, and J. Salmon. Efficient smoothed concomitant lasso estimation for high dimensional regression. *Journal of Physics: Conference Series*, 904(1), 2017.

[OHG09] W. Ou, M. Hämaläinen, and P. Golland. A distributed spatio-temporal EEG/MEG inverse solver. *NeuroImage*, 44(3):932–946, Feb 2009.

[OTJ10] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):231–252, 2010.

[Owe07] A. B. Owen. A robust hybrid of lasso and ridge regression. *Contemporary Mathematics*, 443:59–72, 2007.

[PBC+13] N. Parikh, S. Boyd, E. Chu, B. Peleato, and J. Eckstein. Proximal algorithms. *Foundations and Trends in Machine Learning*, 1(3):1–108, 2013.

[SZ12] T. Sun and C.-H. Zhang. Scaled sparse linear regression. *Biometrika*, 99(4):879–898, 2012.

[Tib96] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 58(1):267–288, 1996.

[TY09] P. Tseng and S. Yun. Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *J. Optim. Theory Appl.*, 140(3):513, 2009.

[van16] S. van de Geer. *Estimation and testing under sparsity*, volume 2159 of *Lecture Notes in Mathematics*. Springer, 2016. Lecture notes from the 45th Probability Summer School held in Saint-Four, 2015, École d'Été de Probabilités de Saint-Flour.

[WD12] J. Wagener and H. Dette. Bridge estimators and the adaptive Lasso under heteroscedasticity. *Math. Methods Statist.*, 21:109–126, 2012.

Image-to-image translation for satellite image time series representation learning

Eduardo H. Sanchez[1,2], Mathieu Serrurier[2], and Mathias Ortner[1]

[1]IRT Saint Exupéry
[2]IRIT, Université Toulouse III - Paul Sabatier

May 29, 2019

Abstract

In this paper, we investigate the image-to-image translation task as a means to learn a suitable representation of satellite image time series in an unsupervised manner. Additionally, we aim to create a disentangled representation of time series: a shared representation that captures the common information between the images of a time series and an exclusive representation that contains the specific information of each image. To address these issues, we propose a model that combines a component called cross-domain autoencoders with the variational autoencoder (VAE) and generative adversarial network (GAN) methods. Satellite image time series provided by the Sentinel-2 mission are used to train our model. By leveraging large amounts of unlabeled data, we show that the obtained representations can be very useful to perform multiple tasks such as image classification, image retrieval and image segmentation.

Keywords: Unsupervised learning, image-to-image translation, VAE, GAN, disentangled representation.

1 Introduction

Deep learning has demonstrated impressive performance on a variety of tasks such as image classification, object detection, semantic segmentation, among others. Typically, these models create internal abstract representations from raw data in a supervised manner. Nevertheless, supervised learning is a limited approach since it requires large amounts of labeled data. It is not always possible to obtain labeled data since it requires time, effort and resources. As a consequence, semi-supervised or unsupervised algorithms have been developed to reduce the required number of labels. Unsupervised learning is intended to learn useful representations of data easily transferable for further usage. Another desirable property of unsupervised methods is to perform dimensionality reduction while keeping the most important characteristics of data. Classical methods are principal component analysis (PCA) or matrix factorization. For the same purpose, autoencoders learn to compress data into a low-dimensional representation and then, to uncompress that representation into the original data. An autoencoder variant is the variational autoencoder (VAE) introduced by Kingma and Welling [1] where the low-dimensional representation is constrained to follow a prior distribution. The VAE provides a way to extract a low-dimensional representation while learning the probability distribution of data. Other unsupervised methods of learning the probability data distribution have been recently proposed using generative models. A generative model of particular interest is generative adversarial networks (GANs) introduced by Goodfellow *et al.* [2, 3].

In this work, we present a model that combines the VAE and GAN methods in order to create a useful representation of satellite image time series in an unsupervised manner. To create these representations we propose to learn the image-to-image translation task introduced by Isola *et al.* [4] and Zhu *et al.* [5]. Given two images from a time series, we aim to translate one image into the other one. Since both images are acquired at different times, the model should learn the common information between these images as well as their differences to perform translation. We also aim to create a disentangled representation into a shared representation that captures the common information between the images of a time series and an exclusive representation that contains the specific information

of each image. For instance, the knowledge about the specific information of each image could be useful to perform change detection.

Since we aim to generate any image of the time series from any of its images, we address the problem of multimodal generation, *i.e.* multiple output images can be generated from a single input image. For instance, an image containing harvested fields could be translated into an image containing growing crop fields, harvested fields or a combination of both.

Our approach is inspired by the BicycleGAN model introduced by Zhu *et al.* [6] to address multimodal generation and the model presented by Gonzalez-Garcia *et al.* [7] to address representation disentanglement.

In this work, the following contributions are made. First, we propose a model that combines the cross-domain autoencoder principle proposed by Gonzalez-Garcia *et al.* [7] under the GAN and VAE constraints to address representation disentanglement and multimodal generation. Our model is adapted to satellite image time series analysis using a simple architecture. Second, we show that our model is capable to process a huge volume of high-dimensional data such as Sentinel-2 image time series in order to create feature representations. Third, our model generates a disentangled representation that isolates the common information of the entire time series and the exclusive information of each image. Finally, our experiments suggest that these feature representations are useful to perform several tasks such as image classification, image retrieval and image segmentation.

2 Related work

Variational autoencoder (VAE). In order to estimate the data distribution of a dataset, a common approach is to maximize the log-likelihood function given the samples of the dataset. A lower bound of the log-likelihood is introduced by Kingma and Welling [1]. To learn the data distribution, the authors propose to maximize the lower bound instead of the log-likelihood function which in some cases is intractable. The model is implemented using an autoencoder architecture and trained via a stochastic gradient descent method. It is an interesting method since it creates a low-dimensional representation where relevant attributes of data are captured.

Generative adversarial networks (GAN). Due to its great success in many different domains, GANs [2, 3] have become one of the most important research topics. The GAN model can be thought of as a game between two players: the generator and the discriminator. In this setting, the generator aims to produce samples that look like drawn from the same distribution as the training samples. On the other hand, the discriminator receives samples to determine whether they are real (dataset samples) or fake (generated samples). The generator is trained to fool the discriminator by learning a mapping function from a latent space which follows a prior distribution to the data space. However, traditional GANs (DCGAN [8], LSGAN [9], BEGAN [10], WGAN [11], WGAN-GP [12], LaplacianGAN [13], EBGAN [14], among others) do not provide a means to learn the inverse mapping from the data space to the latent space. To solve this problem, several models were proposed such us BiGAN [15] or VAE-GAN [16] which include an encoder from the data space to the latent space in the model. The data representation obtained in the latent space via the encoder can be used for other tasks as shown by Donahue *et al.* [15].

Image-to-image translation. It is one of the most popular applications using conditional GANs [17]. The image-to-image translation task consists of learning a mapping function between an input image domain and an output image domain. Impressive results have been achieved by the pix2pix [4] and cycleGAN [5] models. Nevertheless, most of these models are monomodal. That is, there is a unique output image for a given input image.

Multimodal image-to-image translation. One of the limitations of previous models is the lack of diversity of generated images. Certain models address this problem by combining the GAN and VAE methods. On the one hand, GANs are used to generate realistic images while VAE is used to provide diversity in the output domain. Recent work that deals with multimodal output is presented by Gonzalez-Garcia *et al.* [7], Zhu *et al.* [6], Huang *et al.* [**?**], Lee *et al.* [18] and Ma *et al.* [19]. In particular, to be able to generate an entire time series from a single image, we adopt the principle of the BicycleGAN model proposed by Zhu *et al.* [6] where a low-dimensional latent vector represents the diversity of the output domain. However, while the BicycleGAN model is mainly focus on image generation, we only consider the image-to-image translation task as a way to learn suitable feature representations. For image generation purpose, the output diversity is conditioned at the encoder input level in the BicycleGAN model. Instead the output diversity is conditioned at the decoder input level in our model.

Disentangled feature representation. Recent work is focused on learning disentangled representations by isolating the factors of variation of high-

dimensional data in an unsupervised manner. A disentangled representation can be very useful for several tasks that require knowledge of these factors of variation. Chen *et al.* [20] propose an objective function based on the maximization of the mutual information. Gonzalez-Garcia *et al.* [7] propose a model based on VAE-GAN image translators and a novel network component called cross-domain autoencoders. This model separates the representation of two image domains into three parts: the shared part which contains common information from both domains and the exclusive parts which only contain factors of variation that are specific to each domain. We propose a model that combines the cross-domain autoencoder component under the VAE and GAN constraints in order to create representations containing the common information of the entire time series and the exclusive information of each image. The VAE is used to create a low-dimensional representation that encodes the image variations related to acquisition time and the GAN is used to evaluate the generated image at a given time. We introduce a model adapted for satellite image time series using a simpler architecture since only four functions (the shared representation encoder, the exclusive representation encoder, the decoder and the discriminator) must be learned instead of ten functions as in the model [7].

3 Method

Let X, Y be two images from a given time series T in a region C. Let $\mathcal{X}$ be the image domain where these images belong to and let $\mathcal{R}$ be the representation domain of these images. The representation domain $\mathcal{R}$ is divided into two subdomains $\mathcal{S}$ and $\mathcal{E}$, $\mathcal{R} = [\mathcal{S}, \mathcal{E}]$. The subdomain $\mathcal{S}$ contains the common information between images X and Y and the subdomain $\mathcal{E}$ contains the particular information of each image. Since images X and Y belong to the same time series, their shared representations must be identical, *i.e.* $S_X = S_Y$. On the other hand, as images are acquired at different times, their exclusive representations E_X and E_Y correspond to the specific information of each image.

We propose a model that learns the transition from X to Y as well as the inverse transition from Y to X. In order to accomplish this, an autoencoder-like architecture is used. In Figure 1, an overview of the model can be observed. Let $E_{sh} : \mathcal{X} \rightarrow \mathcal{S}$ be the shared representation encoder and $E_{ex} : \mathcal{X} \rightarrow \mathcal{E}$ be the exclusive representation encoder. To generate the image Y, the shared representation of X, *i.e.* $E_{sh}(X)$, and the exclusive representation of Y, *i.e.* $E_{ex}(Y)$ are computed. Then both representations are passed through the decoder function $G : \mathcal{R} \rightarrow \mathcal{X}$ which generates a reconstructed image $G(E_{sh}(X), E_{ex}(Y))$. A similar process is followed to reconstruct the image X. Then, these images are passed through a discriminator function $D : \mathcal{X} \rightarrow [0, 1]$ in order to evaluate the generated images.

The model functions E_{ex}, E_{sh}, G and D are represented by neural networks with parameters $\theta_{E_{ex}}$, $\theta_{E_{sh}}$ and θ_G and θ_D, respectively. The training procedure to learn these parameters is explained below.

3.1 Objective function

As the work presented by Zhu *et al.* [6] and Gonzalez-Garcia *et al.* [7], our objective function is composed of several terms in order to obtain a disentangled representation.

Concerning the shared representation, images X and Y must have identical shared representations, *i.e.* $E_{sh}(X) = E_{sh}(Y)$. A simple solution is to minimize the L_1 distance between their shared representations as can be seen in Equation 1.

$$L_1^{sh} = \mathbb{E}_{X,Y \sim \mathcal{X}} \left[|E_{sh}(X) - E_{sh}(Y)| \right] \tag{1}$$

The exclusive representation must only contain the particular information that corresponds to each image. To enforce the disentanglement between shared and exclusive representations, we include a reconstruction loss in the objective function where the shared representations of X and Y are switched. The loss term corresponding to the reconstruction of image X is represented in Equation 2. Moreover, this loss term can be thought of as the reconstruction loss in the VAE model [1] which maximizes a lower bound of the log-likelihood function. As we enforce representation disentanglement, we simultaneously maximize the log-likelihood function which is equivalent to Kullback-Leibler divergence minimization between the real distribution and the generated distribution.

$$L_1^{X,Y} = \mathbb{E}_{X,Y \sim \mathcal{X}} \left[|X - G(E_{sh}(Y), E_{ex}(X))| \right] \tag{2}$$

On the other hand, the lower bound proposed in the VAE model constraints the feature representation to follow a prior distribution. In our model, we only force the exclusive representation to be distributed as a standard normal distribution $\mathcal{N}(0, I)$ in order to generate multiple outputs by sampling from this space during inference. In contrast to the approach employed by Gonzalez-Garcia *et al.* [7] which uses a GAN approach to constraint the exclusive representation distribution,

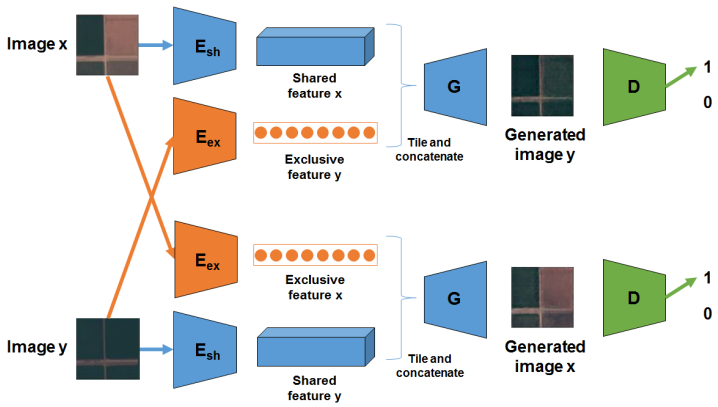


Figure 1: Model overview. The model goal is to learn both image transitions: $X \rightarrow Y$ and $Y \rightarrow X$. Both images are passed through the network E_{sh} in order to extract their shared representations. On the other hand, the network E_{ex} extracts the exclusive representations corresponding to images X and Y. The exclusive representation encoder output is constrained to follow a standard normal distribution. In order to generate the image Y, the decoder network G takes the shared representation of image X and the exclusive representation of image Y. A similar procedure is performed to generate the image X. Finally, the discriminator D is used to evaluate the generated images.

a simpler solution which proves to be effective is to include a Kullback-Leibler divergence term between the distribution of the exclusive representation and the prior $\mathcal{N}(0, I)$. Assuming that the exclusive representation encoder $E_{ex}(X)$ is distributed as a normal distribution $\mathcal{N}(\mu_{E_{ex}(X)}, \sigma_{E_{ex}(X)})$, the Kullback-Leibler divergence can be written as follows

$$L_{KL}^{X} = -\tfrac{1}{2}\mathbb{E}_{X\sim\mathcal{X}}\left[1 + \log(\sigma^2_{E_{ex}(X)}) - \mu^2_{E_{ex}(X)} - \sigma^2_{E_{ex}(X)}\right] \quad (3)$$

In order to minimize the distance between the real distribution and the generated distribution of images, a LSGAN loss [9] is included in the objective function. The discriminator is trained to maximize the probability of assigning the correct label to real images and generated images while the generator is trained to fool the discriminator by classifying generated images as real, *i.e.* $D(G(E_{sh}(Y), E_{ex}(X))) \rightarrow 1$. The corresponding loss term for image X and its reconstructed version can be seen in Equation 4 where the discriminator maximizes this term while the generator minimizes it.

$$L_{GAN}^{X} = \mathbb{E}_{X\sim\mathcal{X}}\left[(D(X))^2\right] + \mathbb{E}_{X,Y\sim\mathcal{X}}\left[(1 - D(G(E_{sh}(Y), E_{ex}(X))))^2\right] \quad (4)$$

To summarize, the training procedure can be seen as a minimax game (Equation 5) where the objective function $\mathcal{L}$ is minimized by the generator functions of the model (E_{ex}, E_{sh}, G) while it is maximized by the discriminator D.

$$\min_{E_{ex},E_{sh},G} \max_{D} \mathcal{L} = L_{GAN}^{X} + L_{GAN}^{Y} + \lambda_{L_1}^{sh} L_1^{sh} + \lambda_{L_1}\left(L_1^{X,Y} + L_1^{Y,X}\right) + \lambda_{L_{KL}}(L_{KL}^{X} + L_{KL}^{Y}) \quad (5)$$

Where λ_{L_1}, $\lambda_{L_1}^{sh}$ and $\lambda_{L_{KL}}$ are constant coefficients to weight the loss terms.

3.2 Implementation

Network architectures: Our model is architectured around four network blocks: the shared representation encoder, the exclusive representation encoder, the decoder and the discriminator. The shared representation encoder is composed of 5 convolutional layers while the exclusive representation encoder is composed of a first convolutional layer and three consecutive ResNet blocks [21]. Since the exclusive representation encoder must provide a normally distributed vector, 2 fully-connected layers of size 64 are appended on top of the ResNet blocks to estimate its mean value μ and standard deviation σ. The decoder consists of 4 transposed convolutional layers. Finally, the discriminator is composed of 5 convolutional layers. During training and

test experiments, we use batch normalization in all the networks. All the layers use a kernel of size 4×4, a stride of 2 and leaky ReLU as activation function (except the discriminator and the decoder outputs where the sigmoid and hyperbolic tangent functions are used, respectively).

Optimization setting: To train our model, we use batches of 64 randomly selected pairs of images of size $64 \times 64 \times 4$ from our time series dataset. Every network is trained from scratch by using randomly initialized weights as starting point. The learning rate is implemented as a staircase function which starts with an initial value of 0.0002 and decays every 50000 iterations. We use Adam optimizer to update the network weights using a $\beta = 0.5$ during 150000 iterations. Concerning the loss coefficients, we use the following values: $\lambda_{L_1} = 10$, $\lambda_{L_1}^{sh} = 0.5$ and $\lambda_{L_{KL}} = 0.01$ during training. The training procedure is summarized in Algorithm 1.

Algorithm 1 Training algorithm.

1: Random initialization of model parameters
2: $(\theta_D^{(0)}, \theta_{E_{sh}}^{(0)}, \theta_{E_{ex}}^{(0)}, \theta_G^{(0)})$
3: **for** $k = 1$; $k = k + 1$; $k <$ number of iterations **do**
4: Sample a batch of m time series $\{T_s^{(1)}, ..., T_s^{(m)}\}$
5: Sample a batch of m image pairs
6: $\{(X^{(1)}, Y^{(1)}), ..., (X^{(m)}, Y^{(m)})\}$ from $\{T_s^{(i)}\}$
7: Compute $\mathcal{L}^{(k)}(X^{(i)}, Y^{(i)}, \theta_D^{(k)}, \theta_{E_{sh}}^{(k)}, \theta_{E_{ex}}^{(k)}, \theta_G^{(k)})$

$$\begin{aligned}\mathcal{L}^{(k)} = \frac{1}{m}\sum_{i=1}^{m}\Big[&\Big(D(X^{(i)})\Big)^2 + \Big(D(Y^{(i)})\Big)^2 \\ &+ \Big(1 - D(G(E_{sh}(Y^{(i)}), E_{ex}(X^{(i)})))\Big)^2 \\ &+ \Big(1 - D(G(E_{sh}(X^{(i)}), E_{ex}(Y^{(i)})))\Big)^2 \\ &+ \lambda_{L_1}\Big(|X^{(i)} - G(E_{sh}(Y^{(i)}), E_{ex}(X^{(i)}))| \\ &+ |Y^{(i)} - G(E_{sh}(X^{(i)}), E_{ex}(Y^{(i)}))|\Big) \\ &+ \lambda_{L_1}^{sh}\Big(|E_{sh}(X^{(i)}) - E_{sh}(Y^{(i)})|\Big) \\ &- \frac{1}{2}\lambda_{L_{KL}}\Big(2 + \log(\sigma^2_{E_{ex}(X^{(i)})}) - \mu^2_{E_{ex}(X^{(i)})} \\ &- \sigma^2_{E_{ex}(X^{(i)})} + \log(\sigma^2_{E_{ex}(Y^{(i)})}) - \mu^2_{E_{ex}(Y^{(i)})} \\ &- \sigma^2_{E_{ex}(Y^{(i)})}\Big)\Big]\end{aligned} \tag{6}$$

8: Update the model parameters:

$$\theta_D^{(k+1)} \leftarrow \text{Adam}\left(-\nabla_{\theta_D^{(k)}}\mathcal{L}^{(k)}, \theta_D^{(k)}\right) \tag{7}$$

$$\theta_{E_{sh}}^{(k+1)} \leftarrow \text{Adam}\left(\nabla_{\theta_{E_{sh}}^{(k)}}\mathcal{L}^{(k)}, \theta_{E_{sh}}^{(k)}\right) \tag{8}$$

$$\theta_{E_{ex}}^{(k+1)} \leftarrow \text{Adam}\left(\nabla_{\theta_{E_{ex}}^{(k)}}\mathcal{L}^{(k)}, \theta_{E_{ex}}^{(k)}\right) \tag{9}$$

$$\theta_G^{(k+1)} \leftarrow \text{Adam}\left(\nabla_{\theta_G^{(k)}}\mathcal{L}^{(k)}, \theta_G^{(k)}\right) \tag{10}$$

9: **end for**

4 Experiments

4.1 Sentinel-2

The Sentinel-2 mission is composed of a constellation of 2 satellites that orbit around the Earth providing an entire Earth coverage every 5 days. Both satellites acquire images at 13 spectral bands using different spatial resolutions. In this paper, we use the RGBI bands which correspond to bands at 10m spatial resolution. In order to organize the data acquired by the mission, Earth surface is divided into square tiles of approximately 100 km on each side. One tile acquired at a particular time is referred to as a granule.

To create our dataset, we selected 42 tiles containing several regions of interest such as the Amazon rainforest, the Dead Sea, the city of Los Angeles, the Great Sandy Desert, circular fields in Saudi Arabia, among others. As explained by Kempeneers and Soille [22], many of the acquired granules might carry useless information. In our case, the availability of granules for a given tile depends on two factors: the cloud coverage and the image completeness. Therefore, we defined a threshold in order to avoid these kind of problems that affect Earth observation by setting a cloud coverage tolerance of 2% and completeness tolerance of 85%. For each tile, we extracted 12 granules from March 2016 to April 2018. Then, we selected 25 patches of size 1024×1024 from the center of the tiles to reduce the effect of the satellite orbit view angle. Finally, our dataset is composed of 1050 times series each of which is composed of 12 images of size $1024 \times 1024 \times 4$.

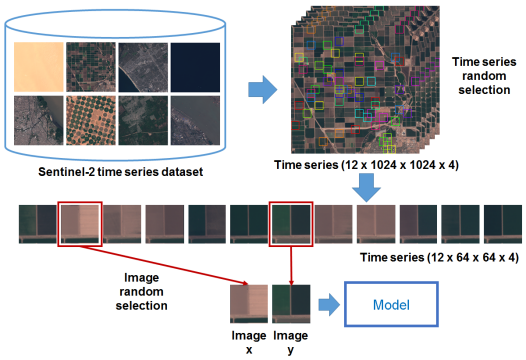


Figure 2: Training data selection. A batch of smaller time series is randomly sampled from the dataset. At each iteration two images are randomly selected from each time series to be used as input for our model.

In order to analyze the entire time series using smaller patches the following strategy is applied: a batch of time series composed of images of size $64 \times 64 \times 4$ is randomly sampled from the time series of size $1024 \times 1024 \times 4$. Since our model takes 2 images as input, at each iteration two images are randomly selected from the time series to be used as input for our model. Thus, the whole time series is learned as the training procedure progresses. Data sampling procedure is depicted in Figure 2.

To evaluate the model performance and the learned representations, we perform several supervised and unsupervised experiments on Sentinel-2 data as suggested by Theis in [23]. We evaluate our model on: a) image-to-image translation to validate the representation disentanglement and b) image retrieval, image classification and image segmentation to validate the shared representation.

We use the code provided by Gonzalez-Garcia *et al.* [7] to run their model on the same dataset of Sentinel-2 data. Unfortunately, the model fails to converge. As a consequence, it was not possible to evaluate the learned features and compare the performance on the proposed tasks with respect to our method.

4.2 Image-to-image translation

It seems natural to first test the model performance at image translation. We use a test dataset which is composed of time series acquired from different tiles to guarantee that training and test datasets are independent. For each dataset, 150 batches of 64 time series are randomly selected. It represents around 20k processed images of size $64 \times 64 \times 4$.

An example of image-to-image translation can be observed in Figure 3. For instance, let us consider the image in the third row, fifth column. The shared representation is extracted from an image X which corresponds to growing crop fields while the exclusive representation is extracted from another image Y where fields have been harvested. Consequently, the generated image contains harvested fields which is defined by the exclusive representation of image Y. In general, generated images look realistic in both training and test datasets except for small details which are most likely due to the absence of skip connections in the generator. We do not include them in our model as they can prevent the shared and exclusive representations to capture the image information which could leak through the skip connections.

We quantify the L_1 distance between generated images $G(E_{sh}(X), E_{ex}(Y))$ and images Y used to extract the exclusive representation. Results can be observed in Table 1 (first row). Pixel values in generated images and real images are in the range of $[-1, 1]$, thus a mean difference of 0.0152 in the training dataset indicates that the model performs well at image-to-image translation. A slightly difference is obtained in the test dataset where the L_1 distance is 0.0207.

A special image-to-image translation case is image autoencoding where the shared and exclusive representations are extracted from the same image. Additionally, we compute the L_1 distance between images X and autoencoded images $G(E_{sh}(X), E_{ex}(X))$ for comparison purpose in Table 1 (second row). Lower values in terms of L_1 distance are obtained with respect to those of image-to-image translation. It is important to note that input images are considerably well reconstructed even if this case is not considered during training. Finally, we perform times series reconstruction in order to show that the exclusive representation encodes the specific information of each image. An image is randomly selected from a time series to extract its shared representation. While keeping the shared representation constant and only modifying the exclusive representation, we reconstruct the original time series. Results in terms of L_1 distance between the original time series and the reconstructed one can be observed in Table 1 (third row). Similar values to those of image-to-image translation are obtained. An example of time series reconstruction can be seen in Figure 4.

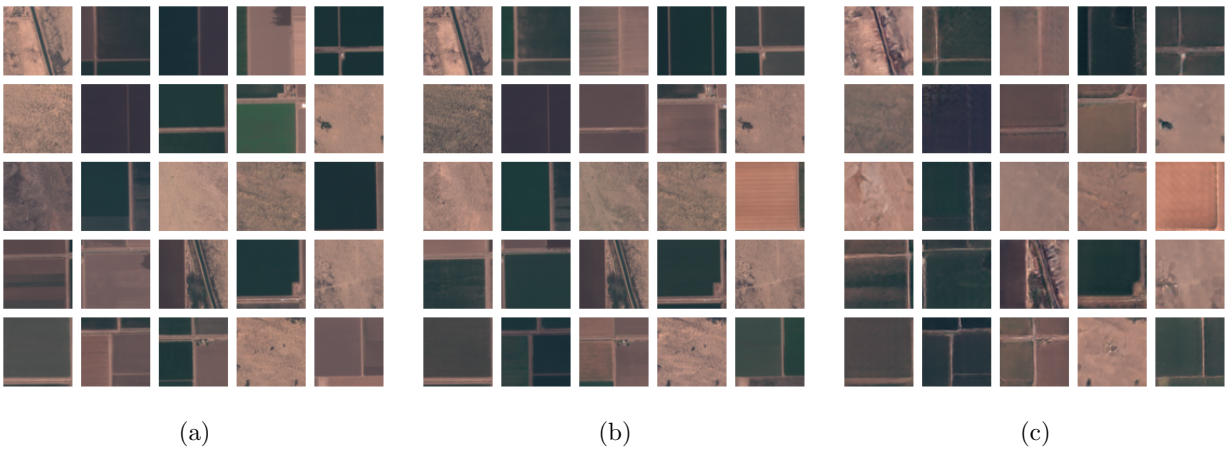


Figure 3: Image translation performed on images of Brawley, California. (a) Images used to extract the shared representations; (b) Images used to extract the exclusive representations; (c) Generated images from the shared representation of (a) and the exclusive representation of (b).

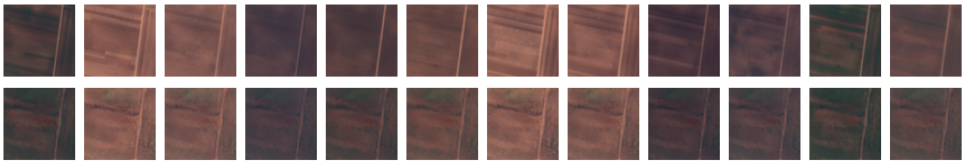


Figure 4: Multimodal generation. The first row corresponds to a time series sampled from the test dataset. The second row corresponds to a time series where each image is generated by using the same shared representation and only modifying the exclusive representation.

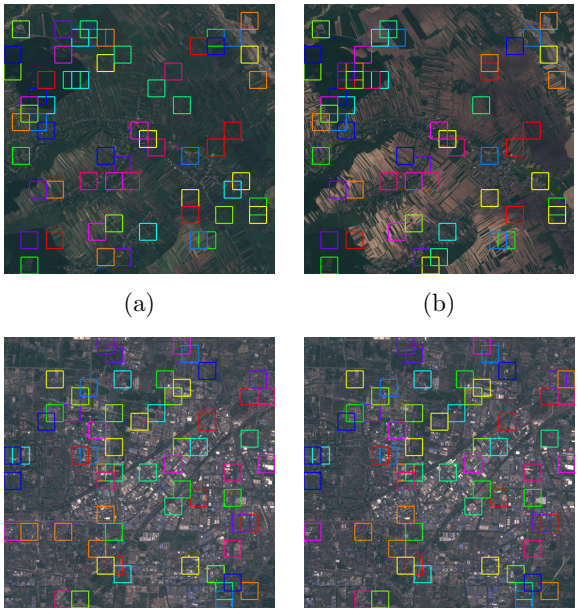
Task	Training set	Test set
Image-to-image	0.0152 ± 0.0573	0.0207 ± 0.0774
Autoencoding	0.0084 ± 0.0309	0.0087 ± 0.0312
Time series	0.0177 ± 0.0622	0.0223 ± 0.0828

Table 1: Mean and standard deviation values of the L_1 distance for image-to-image translation (first row), image autoencoding (second row) and time series reconstruction (third row).

4.3 Image retrieval

In this experiment, we want to evaluate whether the shared representation provides information about the geographical location of time series via image retrieval. Given an image patch from a granule acquired at time t_o, we would like to locate it in a granule acquired at time t_f. The procedure is the following: a time series of size $12 \times 1024 \times 1024 \times 4$ is randomly sampled from the dataset. Then, a batch of 64 image patches of size $64 \times 64 \times 4$ is randomly selected as shown in Figure 5a. The corresponding shared representations are extracted for each image of the batch. The main idea is to use the information provided by the shared representation to locate the image patches in every image of the time series. For each image of the time series, a sliding window of size $64 \times 64 \times 4$ is applied in order to explore the entire image. As the window slides, the shared representations are extracted and compared to those of the images to be retrieved. The nearest image in terms of L_1 distance is selected as the retrieved image at each image of the time series. In our experiment, 150 time series of size $12 \times 1024 \times 1024 \times 4$ are analyzed. It represents around 115k images of size $64 \times 64 \times 4$ to be retrieved and 110M images of size $64 \times 64 \times 4$ to be analyzed.

To illustrate the retrieval algorithm, let us consider



(a) (b) (c) (d)

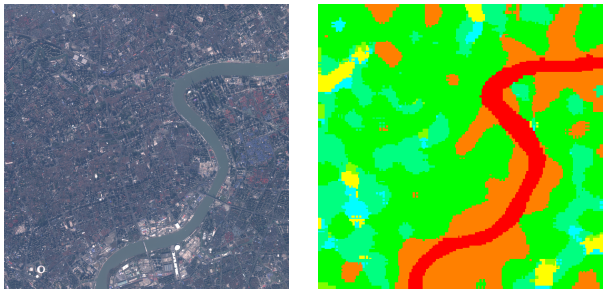
Figure 5: Image retrieval using shared representation comparison. (a)(c) Selected image from a time series where a batch of 64 patches (colored boxes) are extracted from; (b)(d) Another image from the same time series is used to locate the selected patches. The algorithm plots colored boxes corresponding to the nearest patches in terms of shared representation distance.

a test image of agricultural fields. We plot the patches to be retrieved in Figure 5a and the retrieved patches by the algorithm in Figure 5b. As can be seen, even if some changes have occurred, the algorithm is able to spatially locate most of the patches. In spite of the seasonal changes in the agricultural fields, the algorithm performs correctly since the image retrieval leverages the shared representation which contains common information of the time series. Results in terms of Recall@1 are displayed in Table 2 (first row). We obtain a high value in terms of Recall@1 even if it is not so close to 1. This result can be explained since the dataset contains several time series from the desert, forest and ocean tiles which could be notoriously difficult to retrieve even for humans. For instance, image retrieval performs better in urban scenarios since the city provides details that can be easily identified (see Figures 5c and 5d) in contrast to agricultural fields where distinguishing textures can be confusing.

As a baseline to compare to the retrieval image based on the shared representations, we use the raw pixels of the image to find the image location. Our experiments show that using raw pixels as feature yields a poor performance to locate the patches (see Table 2, second row). We note that even if the retrieved images look similar to the query images, they do not come from the same location. The recommended images using raw pixels are mainly based on the image color. Whenever a harvest fields is used as query image the retrieved images correspond to harvested fields as well. This is not the case when using shared representations since seasonal changes are ignored in the shared representation.

Method	Recall@1
Shared representations	0.7372
Raw pixels	0.5083

Table 2: Image retrieval results in terms of Recall@1 using the shared representation and the raw pixels of the image as feature.



(a) (b)

Figure 6: Image segmentation in Shanghai, China. A sliding window is used to extract the shared representations of the image which in turn are used to perform clustering with 7 classes. (a) Image to be segmented; (b) Segmentation map.

4.4 Image classification

A common method to evaluate the performance of unsupervised features is to apply them to perform image classification. We test the shared representations extracted by our model using a novel dataset called EuroSAT [24]. It contains 27000 labeled images in 10 classes (residential area, sea, river, highway, etc.). We divide the dataset into a training and test dataset using

a 80:20 split keeping a proportional number of examples per class.

We recover the shared representation encoder $E_{sh}(\cdot)$ as feature extractor from the pretrained model. We append two fully-connected layers of 64 and 10 units, respectively on top of the feature extractor. We only train these fully-connected layers while keeping frozen the weights of the feature extractor in a supervised manner using the training split of EuroSAT. Results can be observed in Table 3. We obtain an accuracy of 92.38% while we achieve an accuracy of 94.54% by not freezing the weights of the feature extractor during training. It is important to note that using pretrained weights reduces the training time and allows to achieve better performance with respect to randomly initialized weights (62.13% of accuracy after 50 epochs).

Model	Accuracy	Epochs
Pretrained + Fine-tuning	92.38%	10
Pretrained + Full-training	94.54%	10
From scratch	62.13%	50

Table 3: Accuracy results in the test dataset from classification experiments.

Higher accuracy (98.57%) in the EuroSAT dataset is claimed by Helber *et al.* [24] using supervised pretrained GoogLeNet or ResNet-50 models. However, we show that using a very simple model trained in an unsupervised manner allows us to obtain excellent results.

4.5 Image segmentation

Since the shared representation is related to the location and texture of the image, we perform a qualitative experiment to illustrate its use for image segmentation. As image segmentation is a well-studied task, we do not pretend to achieve state-of-the-art results but we aim to show some properties of the learned representations. An image of size $1024 \times 1024 \times 4$ is randomly selected from a time series. Then, a sliding window of size $64 \times 64 \times 4$ and stride of size 32×32 is used to extract the patches. The shared representations extracted from these patches are used to perform clustering via k-means with N clusters defined by the user. A new sliding window with a stride of 8×8 is used to extract the shared representations from the image. Each of the extracted shared representations is assigned to a cluster. Since several clusters are assigned for each pixel, the cluster is decided by the majority of voted clusters. In Figure 6, a segmentation map example in Shanghai is displayed. As can be seen, this unsupervised image segmentation method achieves interesting results. It is able to segment the river, the port area and the residential area, among others. We think that segmentation results can be improved by using a smaller windows to achieve better resolution. On the other hand, experiments using the raw pixels of the image as features produce segmentation maps of lower visual quality.

5 Conclusion

In this work, we investigate how to obtain a suitable data representation of satellite image time series. We first present a model based on VAE and GAN methods combined with the cross-domain autoencoder principle. This model is able to learn a disentangled representation that consists of a common representation for the images of the same time series and an exclusive representation for each image. We train our model using Sentinel-2 time series which indicates that the model is able to deal with huge amounts of high-dimensional data. Finally, we show experimentally that the disentangled representation can be used to achieved interesting results at multiple tasks such as image classification, image retrieval and image segmentation. While this work is mainly focused on the use of the shared representation to perform downstream tasks, we leave the use of the exclusive representation for future work. Another possible avenue of research is to include the temporal information of time series as our model does not take into account the chronological order when sampling an image pair. We think that including time information could be useful for change detection.

References

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, 2014.

[3] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. 2016.

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with

conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[5] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[6] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems 30*.

[7] Abel Gonzalez-Garcia, Joost van de Weijer, and Yoshua Bengio. Image-to-image translation for cross-domain disentanglement. In *Advances in Neural Information Processing Systems*. 2018.

[8] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

[9] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*.

[10] David Berthelot, Tom Schumm, and Luke Metz. BEGAN: boundary equilibrium generative adversarial networks. 2017.

[11] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[12] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein GANs. In *Advances in Neural Information Processing Systems*, 2017.

[13] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, 2015.

[14] Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. Energy-based generative adversarial network. In *International Conference on Learning Representations*, 2017.

[15] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *International Conference on Learning Representations*, 2017.

[16] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[17] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. 2014.

[18] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

[19] Liqian Ma, Xu Jia, Stamatios Georgoulis, Tinne Tuytelaars, and Luc Van Gool. Exemplar guided unsupervised image-to-image translation. *arXiv preprint arXiv:1805.11145*, 2018.

[20] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2016.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] P Kempeneers and P Soille. Optimizing Sentinel-2 image selection in a big data context. *Big Earth Data*, 1(1-2):145–158, 2017.

[23] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, 2016.

[24] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. 2017.

Filtrage collaboratif explicite par analyse de sentiments à l'aveugle

Auteur A*[1] et Auteur B[2]

[1]Université X, CNRS
[2]Université Y, CNRS et INRIA

10 avril 2019

Résumé

En recommandation, les modèles de filtrage collaboratif sont souvent qualifiés de boites noires. Pour palier ce problème, nous proposons de mettre en parallèle analyse de sentiments et filtrage collaboratif dans un schéma que nous qualifions "d'analyse de sentiments à l'aveugle". Concrètement, nous proposons, par apprentissage adverse adverse d'effectuer du filtrage collaboratif par génération de textes plutôt que via la prédiction directe de notes. Nous détaillons comment ce schéma permet, en plus d'un gain de performances, d'expliciter naturellement la recommandation.

Mots-clef : Filtrage Collaboratif, Réseaux de Neurones, Réseaux Adverses.

1 Introduction

En recommandation, les modèles de l'état de l'art étant majoritairement appris à l'aide d'interactions utilisateur-produit, ils sont souvent abstrait [KBV09]. De ce fait, les algorithmes de filtrage collaboratif manquent de transparence et sont souvent qualifiés de boites-noires [BeK14]. En effet, les seules explications proviennent généralement d'une interprétation du voisinage et sont limitées à de simples comparaisons entre produits : *nous vous proposons XXX parce que vous avez aimé YYY*. Pourtant, il est crucial qu'une suggestion soit explicité. De plus, pour inspirer confiance, un système de recommandation se doit d'être transparent [TM07].

Pour améliorer conjointement performance et transparence, il est commun d'exploiter les avis textuels associées aux notes. Une première approche consiste à créer [DGG16, CC17] des profils uniquement textuels, naturellement interprétable. Une seconde approche consiste plutôt à enrichir des profils par l'apprentissage conjoint d'un modèle prédictif et d'un modèle thématique [ML13], de langue [AKCC15, NLVM17] ou d'analyse de sentiments [DdGGG18]. Dans tous les cas, l'enjeu est double : être capable de proposer un texte support – par extraction ou génération – pour chaque suggestion tout en ayant des profils encodant efficacement les gouts des utilisateurs.

Pour obtenir à la fois des profils interprétables et des suggestions explicite, nous explorons ici l'utilisation des réseaux adverses dans le cadre du filtrage collaboratif. Il s'agit d'exploiter les capacités de génération [KLA18] et de désambiguïsassions [CDH+16, LZU+17] de l'espace des modèles adverses pour créer un système de recommandation transparent. Jusqu'à présent, ces techniques ont été uniquement appliquées dans le cadre de la recommandation de séquences [BPL18, HHDC18].

Pour lier texte et note, nous exploitons le cadre de l'analyse de sentiment en parallèle de la recommandation. l'objectif est de tirer profit de la similitude entre ces tâches pour améliorer le filtrage collaboratif. Dans cet article, plutôt que d'associer les tâches, nous les opposons dans un schéma d'apprentissage adverse. En ce sens, nous partons d'une observation simple : l'analyse de sentiment, lorsqu'elle prend en compte les spécificités de chaque utilisateurs et produits [CST+16], est similaire au filtrage collaboratif dans sa version régressive. La seule différence est l'absence de texte. Nous pouvons alors considérer le filtrage collaboratif comme de l'analyse de sentiment "à l'aveugle".

*Si vraiment vous voulez mettre votre email ou page web...: A.A@univ-x.fr

Ce papier s'organise de la façon suivante : Dans un premier temps, nous détaillons l'état de l'art en prédiction de note. Ensuite, nous formalisons le concept d' analyse de sentiment "à l'aveugle" avant de détailler le modèle adverse considéré. Enfin, nous proposons une évaluation quantitative et qualitative de notre modèle.

2 État de l'art

Dans cette section, nous détaillons brièvement les différents concepts abordés dans ce papier : le filtrage collaboratif et les réseaux adverses.

Le filtrage collaboratif se résume concrètement à l'apprentissage de préférences directement à partir d'interactions (clics, notes) d'utilisateurs. L'hypothèse sous-jacente est que si plusieurs personnes ont des intérêts communs pour certaines choses alors ces intérêts s'étendent probablement à d'autres produits. Tout l'enjeux des algorithmes de filtrage collaboratif est d'exploiter les corrélations d'interactions afin d'inférer des similarités existantes entre produits et utilisateurs. Suite au concours Netflix [BL07], les méthodes de filtrage collaboratif par factorisation se sont imposées [KBV09, SMH07]. Il s'agit d'obtenir des profils en décomposant la matrice d'interactions en deux sous matrices ; une codant pour les utilisateurs, une pour les produits. Aujourd'hui, la plupart des algorithmes développés utilisent des réseaux de neurones [HKBT15, HLZ+17] qui, grâce à une grande flexibilité de modélisation permettent la prise en compte d'un nombre croissant de facteurs. Pourtant, malgré cette prise en compte d'un vaste nombre d'informations et l'amélioration constante de la précision les systèmes de recommandations, ils sont encore souvent qualifiées de "boites noires" [BeK14].

Pour expliciter les suggestions des systèmes de recommandation, une technique consiste à exploiter les avis textuels en plus des notes. Concrètement, il s'agit, en plus des suggestions, de proposer un texte support, détaillant – de manière personnalisée – les avantages d'un produit. Une première approche consiste à simplement utiliser le texte comme un objectif auxiliaire. [AKCC15, NLVM17] proposent de directement prédire l'avis en plus de la note à l'aide d'un réseau de neurones récurrents. En pratique, en plus de permettre de régulariser les profils appris [AKCC15], cela permet d'expliciter les suggestions émises par le système. A l'inverse, [ZNY17, CC17, DGG16] proposent de construire directement chaque profils utilisateur/produit à partir de l'ensemble de ses avis textuels tout en apprenant une affinité textuel. Enfin, [DdGGG18] propose de lier analyse de sentiment et filtrage collaboratif afin d'extraire les biais textuels. Ces modèles permettent, par extraction, d'obtenir du texte support.

Les réseaux adverses ont permis d'obtenir de très bon résultats sur une multitudes de tâches génératives. Ces modèles s'inspirent de la théorie des jeux : un générateur et un discriminateur s'améliorent au cours du temps grâce à une compétition.

Ces deux modèles sont entraînés dans un jeu à somme nul : $\mathcal{G}$ apprend à tromper $\mathcal{D}$ qui lui essaye d'apprendre justement à ne pas être trompé. Dans leurs formulation originale, il s'agit de d'entraîner parallèlement :

— le discriminateur $\mathcal{D}$ à faire la différence entre les vrais et faux examples en maximisant à la fois $\mathbb{E}_{x\sim p_r(x)}[\log \mathcal{D}(x)]$ et $\mathbb{E}_{z\sim p_z(z)}[\log(1-\mathcal{D}(\mathcal{G}(z)))]$
— le générateur $\mathcal{G}$ à minimiser la capacité de $\mathcal{D}$ à repérer ses exemples $\mathbb{E}_{z\sim p_z(z)}[\log(1-\mathcal{D}(\mathcal{G}(z)))]$

Il convient de noter que les réseaux adverses, surtout dans leurs formulation d'origine, sont notoirement difficile à entraîner [[GPAM+14]. Ces modèles sont particulièrement sensibles aux hyper-paramètres et à l'équilibre générateur/discriminateur. Cependant, leur principal intérêt réside dans leurs capacité à générer des données ayant l'air *a priori* vraie [KLA18]. Ces modèles sont principalement utilisés en image, domaines où leurs résultats sont les plus spectaculaires. Dans le cas du texte, l'espace étant discret, il est plus compliqué d'apprendre des GAN. Néanmoins, certaines extensions existes [FGD18]. De plus, contrairement aux modèles à base de réseaux de neurones récurrents qui n'offrent aucunes garanties, l'intérêt principal des réseaux adverses est que, dans certaines conditions, l'espace appris est non-ambiguë [LZU+17, CDH+16]. Cette propriété permet de retrouver certain facteurs latents, comme la présence où l'absence de lunettes par exemple, rendant l'interprétation directe.

Dans le cadre de ces travaux, c'est ces deux capacités de génération et de désambiguïsation qui nous intéresse. Celles-ci permettraient respectivement de rendre les système de recommandation plus explicite et transparent. En effet, il serait à la fois possible de générer des explications tout en pouvant interpréter pourquoi celles-ci ont été générées.

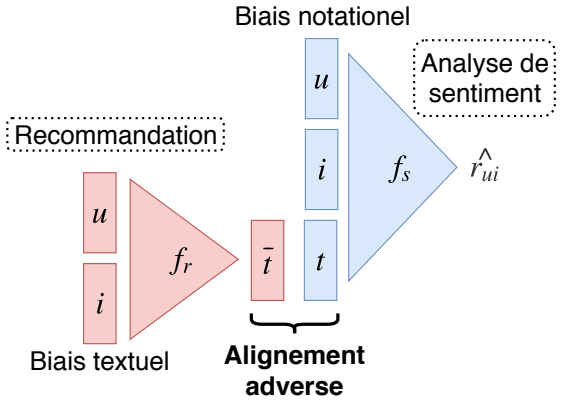


FIGURE 1 – Représentation schématique du concept d'analyse de sentiment à l'aveugle. Le modèle de recommandation (f_r, à gauche) a pour objectif de prédire l'avis t utilisé par le modèle d'analyse de sentiment (f_s, à droite) dans la prédiction de la note r_{ui} de l'utilisateur u sur le produit i. L'enjeu pour f_r est d'arriver à générer un texte $\bar{t}$ qui soit le plus proche possible du texte original t afin que f_s prédise la bonne note $\hat{r_{ui}}$.

3 Analyse de sentiment à l'aveugle

Ici, notre objectif est d'ancrer le filtrage dans un espace latent explicite et non-ambigu. Concrètement, il s'agit d'obtenir à la fois des profils latents – d'utilisateurs et de produits – interprétables et des suggestions explicites. Dans ce papier, pour atteindre cet objectif, nous proposons de mettre en compétition analyse de sentiments et filtrage collaboratif dans le cadre de l'apprentissage adverse. En effet, en plus de leurs capacités génératives [KLA18], les réseaux adverses permettent d'obtenir – en fonction de leurs formulations – des espaces non-ambigüs [CDH+16, LZU+17].

Pour mettre le filtrage collaboratif dans un cadre d'apprentissage adverse nous le considérons comme de l'analyse de sentiments "à l'aveugle" (fig. 1). En pratique, la seule différence entre analyse de sentiment et filtrage collaboratif est l'absence de texte, les avis utilisateurs n'existant pas a priori. Nous proposons alors d'utiliser – comme modèle de recommandation – un modèle de prédiction de texte, l'apprentissage adverse servant à aligner les textes prédits avec les textes réels. Le texte prédit sera utilisé par un modèle d'analyse de sentiments qui devra, en plus de prédire de la polarité du texte, savoir si celui-ci est légitime. Le principal avantage de cette modélisation est que le modèle de génération de texte, comme dans [NLVM17], représente une manière intégrée d'interpréter les suggestions du modèle.

Concrètement, nous proposons de décomposer la tâche de filtrage collaboratif en deux sous-tâches : la prédiction d'un texte $t_{u,i}$, puis de la note associée $r_{u,i}$ (fig. 3). Nous postulons que cette décomposition évitera au modèle de se focaliser sur le problème dit de "complétion de matrice" et sur les biais de notation. En effet, à l'inverse, pour optimiser son objectif le modèle devra générer un texte fidèle à l'original il devrait plutôt encoder les biais d'écriture, ce qui nous intéresse pour expliciter les suggestions. Les biais de notations seront eux captés par le modèle d'analyse de sentiment.

$$(u,i) \xrightarrow{f_r} t_{u,i} \xrightarrow{f_s} r_{u,i}$$

FIGURE 2 – Décomposition du filtrage collaboratif en deux temps : prédiction de texte $t_{u,i}$ puis de note $r_{u,i}$

Le modèle proposé en figure 1 est simplement conceptuel. En pratique, dans la formulation classique des réseau adverses, avoir un générateur paramétrique ne semble pas possible. De plus, la génération de texte via les réseaux adverses – discret par nature – ne fonctionne pas aussi bien que celle des images et reste à l'état de concept. Cependant, dans la section suivante nous proposons une instanciation particulière de ce modèle qui permet de s'abstraire de ces problèmes.

4 Modèle adverse proposé

Formellement soit des tuples utilisateur, produit, note, texte (u, i, r, t) comme données. Nous avons deux tâches : l'analyse de sentiment et le filtrage collaboratif. L'objectif de la première tâche est de prédire r, à l'aide du texte t mais aussi avec les profils u, i. Pour la seconde, l'objectif final est le même ; prédire la note mais seulement avec les profils (u, i). Dans cette contribution, nous proposons de mettre les deux tâches en compétitions : Au lieu de simplement prédire la note directement depuis les profils, nous inférons d'abord un clone $\bar{t}$ du texte original t qui sera utilisé à sa place dans le prédicteur de sentiment. La prédiction d'une note r_{ui} se fera donc de la manière suivante :

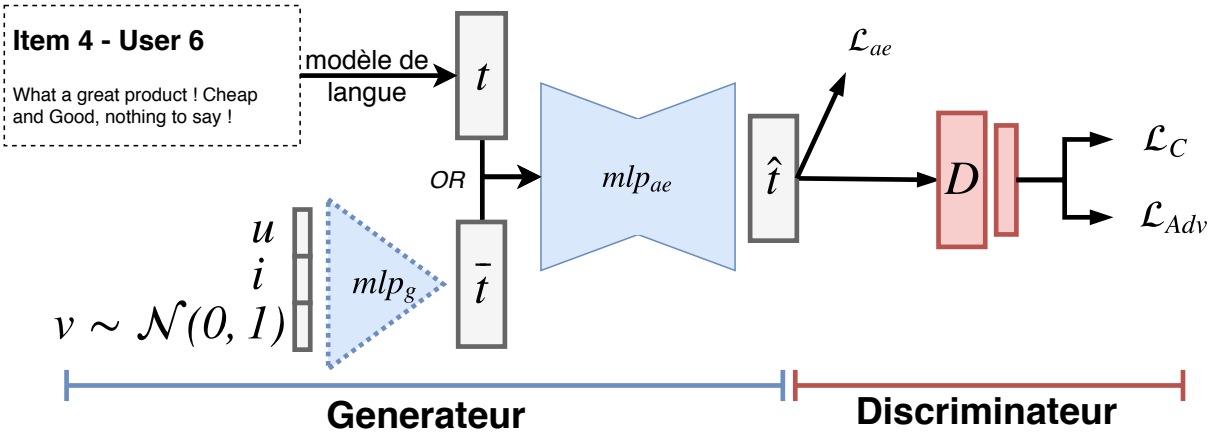


FIGURE 3 – (à gauche) Generateur : (a) Génère une représentation clone $\bar{t}$ à partir d'une paire de représentations (utilisateur,produit). (b) Auto-encode t ou $\bar{t}$ en $\hat{t}$. (à droite) Discriminateur : Predit le sentiment et la source de $\hat{t}$

$$r_{ui} = \underbrace{f_s(\overbrace{f_r(u,i)}^{\text{Recommandation}})}_{\text{Analyse de Sentiment}} \tag{1}$$

Par nature, le texte pose deux problème quant à l'utilisation des réseaux adverses tel que défini par [GPAM+14]. Déjà, il n'est pas utilisable en l'état au sein d'un réseau de neurone et nécessite une vectorisation. De plus, la génération de données discrète empêche le bon apprentissage des générateur. Ici, afin de nous abstraire de ces contrainte et par simplicité, nous proposons dans un premier temps d'encoder le texte à l'aide de *word2vec* [MCCD13]. Concrètement, nous apprenons dans un premier temps des représentations pour chaque mots de l'ensemble d'apprentissage. Ensuite, chaque avis est représenté comme le barycentre de ses représentation de mots :

$$t = \frac{1}{n}\sum_{i=1}^{n} w_i \tag{2}$$

avec $w_i \in \mathbb{R}^d$ les représentations de mot issue de *word2vec*. Par la suite, lorsque nous faisons référence à t, il s'agit donc d'une représentation $t \in \mathbb{R}^d$ et non d'un texte brut.

Notre modèle adverse (fig. 3) est composé de deux sous-reseaux – un générateur et un discriminateur – optimisés à tour de rôle dans un schéma d'apprentissage adverse. Nous détaillons d'abord séparément les fonctions de chaque réseau avant d'expliciter les coûts optimisés.

Générateur : Le premier sous-réseau – f_r – fait office de générateur. Il est lui même composé de deux modules : un auto-encodeur mlp_{ae} et un perceptrons multicouches mlp_g qui génère un texte clone $\bar{t}$ à partir d'un triplet (u, i, v) ; avec v est un vecteur de "bruit". Génerer de but en blanc du texte est complexe. Pour aider mlp_g, l'auto-encodeur apprend la distribution des données en auto-encodant le texte réel t en une représentation $\hat{t}_r$. De ce fait, mlp_g est simplement entraîné à générer un texte "reconstructible" $\hat{t}_f$, indissociable de $\hat{t}_r$; ce texte devant logiquement être t.

$$u, i, v \in \mathcal{R}^d, \qquad \bar{t} = mlp_g(u, i, v), \qquad \hat{t} = mlp_{ae}(t \vee \bar{t}) \tag{3}$$

Formellement, le générateur minimise trois coûts séparés :

($\mathcal{L}_{\mathbf{ae}}$) Tout d'abord, le générateur doit être capable d'auto-encoder un texte originale t. Pour cela, il doit minimiser l'erreur de reconstruction. Ici, nous utilisons l'erreur quadratique comme erreur de reconstruction : $MSE(t, \hat{t}_r)$ [1]. En apprenant à auto-encoder les données réelles, le générateur apprend la distribution des données [GLTFS87]. En parallèle, les clones générés doivent également être "reconstructibles". Nous minimisons

1. $MSE(x, \hat{x}) = (x - \hat{x})^2$

donc également l'erreur de leur reconstruction $MSE(\bar{t}, \hat{t}_f)$. De cette façon, nous garantissons que tous les $\hat{t}$ proviennent de la distribution des données apprises.

($\mathcal{L}^{\mathbf{g}}_{\mathbf{Adv}}$) Cependant, rien ne garantie que les reconstructions de texte généré $\hat{t}_f$, et celles de texte réel $\hat{t}_r$ soit similaire. Pour contraindre le générateur à s'aligner sur les données réelles, les reconstructions doivent être indissociables. Cette propriété est apprise grâce à une optimisation min-max où le générateur et le discriminateur on des objectifs opposés $\mathcal{L}^g_{adv}$ et $\mathcal{L}^d_{adv}$. Concrètement, le générateur, via l'auto-encodeur, va essayer de faire passer la reconstruction du texte réelle $\hat{t}_r$ pour la reconstruction du clone $\hat{t}_f$ et vice-versa. Formellement, le coup minimisée est l'entropie croisée binaire : $BCE(mlp_d(\hat{t}_r), 0)$ et $BCE(mlp_d(\hat{t}_f), 1)$ [2]

($\mathcal{L}_{\mathbf{C}}$) Le sentiment r lié aux reconstuctions $\hat{t}_r$ doit être correctement prédit. Cela ce fait dans le cadre d'une minimisation simultanée de la log-vraisemblance par le générateur.

Finalement, la fonction de coût du générateur est la suivante :

$$\begin{aligned}\mathcal{L}_g &= \mathcal{L}_{ae} + \mathcal{L}^g_{adv} + \mathcal{L}_C \\ &= MSE(t, \hat{t}_r) + MSE(\bar{t}, \hat{t}_f) \\ &+ BCE(mlp_d(\hat{t}_f), 1) + BCE(mlp_d(\hat{t}_r), 0) \\ &+ CCE(mlp_d(\hat{t}_r), r)\end{aligned} \tag{4}$$

Où CCE et BCE veulent respectivement dire Categorical et Binary Cross Entropy.

Discriminateur : Le second sous-réseau – f_s – est composé d'un perceptron multicouches mlp_d à deux têtes. Il fait à la fois office de dicriminateur et de classifieur : Il doit arriver à prédire correctement les polarités associées aux reconstructions $\hat{t}$ tout en arrivant à faire la différence entre celles provenant d'un texte légitime t ou d'un clone $\bar{t}$.

Formellement, le discriminateur minimises donc deux fonctions de coût distinctes :

($\mathcal{L}_{\mathbf{C}}$) Dans un premier temps, le discriminateur, qui fait de l'analyse de sentiment, doit être capable de prédire correctement de la polarité d'une reconstruction $\hat{t}$

($\mathcal{L}^{\mathbf{d}}_{\mathbf{adv}}$) Enfin, a l'inverse du générateur, le discriminateur doit réussir à faire la différence entre une reconstruction issue d'un texte généré $\bar{t}$ et une issue d'un texte original t. Cette compétition entre générateur et discriminateur doit forcer le générateur à mêler $\bar{t}$ et t au sein d'une même reconstruction $\hat{t}$ et, donc, aligner t et $\bar{t}$.

Finalement, avec $\mathcal{L}_C$ l'entropie croisée et $\mathcal{L}^d_{adv}$ la fonction de coût adverse, le coût minimisé est le suivant :

$$\begin{aligned}\mathcal{L}_d &= \mathcal{L}_C + \mathcal{L}^d_{adv}. \\ &= CCE(c, r) + BCE(\hat{t}_f, 0) + BCE(\hat{t}_r, 1)\end{aligned} \tag{5}$$

5 Données, prétraitements & hyperparamètres

Pour rappel, dans cette contribution nous cherchons à exploiter les techniques d'apprentissage adverses avec un double objectif : **(a)** Apprendre les profiles utiles permettant des **(b)** recommandations compréhensibles. Pour évaluer la faculté de notre modèle à atteindre ces deux objectifs, nous proposons plusieurs expériences. Après avoir détaillé les données que nous utilisons pour celles si, nous proposons une évaluation quantitative et qualitative des profils obtenus.

Hyperparametres Lors de nos expériences, l'architecture de notre réseaux est resté constante. Les profils utilisateur, produit ainsi que le vecteur de bruit sont tous de taille identique : $u, v, i \in \mathbb{R}^5$. Ils sont volontairement petit pour eviter le sur-apprentissage. Enfin, la taille des représentation des mots est aussi fixé : emb_size = 100. Les deux fonctions de coûts L_g et L_d sont toutes deux optimisées séquentiellement en utilisant le schéma d'optimisation Adam.

Données et prétraitements Pour nos experiences, nous utilisons des avis consommateurs provenant d'Amazon [MPL15]. Nous choisissons cinq datasets aléatoirement, de différentes tailles, sur différents types d'objets. Les propriétés des bases de données sont répertoriés Table 1. Ici, n'évaluant pas le démarrage à froid, seul les utilisateurs et les produits avec un minimum de 5 avis sont conservés. Les représentations des 10 000 mots les plus fréquents apparaissant au minimum cinq fois sont pré-apprises en utilisant *word2vec*. Afin d'effectuer de la validation croisée, chaque base de donnée est divisé en cinq parts égales. Quatre partie servent pour l'entraînement (80%) et une pour la validation (10%) et l'évaluation (10%). Ce schéma de 80/10/10 est commun dans l'évaluation de système utilisant le texte [ML13, AKCC15].

2. $BCE(x, y) = -[y \cdot \log x + (1 - y) \cdot \log(1 - x)]$

Dataset (#reviews)	Mean (μ)	w/offset	MF	HFT	BSA
Instant Video (37.126)	1.25	1.137	1.024	0.933	**0.924**
Digital Music (64,706)	1.19	0.965	0.903	0.844	**0.832**
Video Games (231,780)	1.45	1.281	1.267	1.097	**1.082**
CSJ (278,677)	1.215	1.323	1.365	1.107	**1.101**
Movie (1,697,533)	1.436	1.148	1.118	1.020	**1.009**

TABLE 1 – Erreur quadratique moyenne en prédiction de note. Les modèles de références sont : La moyenne globale de la base de donnée μ, le bias de notation utilisateur-item global et un algorithme de factorisation matricielle commun [KBV09]. Les valeurs présentées sont des moyennes obtenues par validation croisée sur 5 parties

6 Evaluation en prédiction de note

Nous évaluons dans un premier temps la tâche classique de filtrage collaboratif : la prédiction de note. Cela permet d'évaluer quantitativement les profils appris. Pour prédire la note d'un couple (utilisateur,item) r_{ui} un texte clone $\bar{t}$ est d'abord généré par mlp_g. Ensuite, ce texte est auto-encodé puis classifié. Comme algorithmes de références nous choisissons deux modèles commun : un algorithme de factorisation matricielle [KBV09] et un modèle enrichissant les profils obtenue par factorisation avec le texte - HFT [ML13]. Comme souvent, nous ajoutons les moyennes objet et utilisateur qui sont déjà de bonnes références. L'erreur quadratique moyenne est reportée Table 1.

De façon surprenante, notre modèle, bien que non directement entrainé à optimiser la prédiction de note, obtient de meilleurs performance que les modèles de références qui le sont. Cela confirme, a priori, l'intérêt de modéliser le texte au lieu des notes. Aussi, de tels résultats montre que les profils appris contiennent des informations

7 Explicitation des suggestions

Pour expliciter les suggestions, il est souvent proposé de générer des avis en plus des notes [NLVM17]. Ici, Le modèle présenté n'est pas explicitement entrainé pour cela et il n'est pas capable de générer des textes mots à mots. Cependant, il génère des représentations de mots qui devraient être similaire à la représentation réelle (celle-ci étant une moyenne de mots). Il est donc possible, en comparant la représentation générée $\bar{t}$ avec toutes les représentations de phrases existantes, de créer un avis par extraction. Formellement soit un couple (u, i) cible de la recommandation, $\overline{t_{ui}}$ la représentation généré et $s_{*i} = \{t_{*i}\}$ l'ensemble des représentations des phrases pré-existantes sur l'item i. En utilisant la similarité cosinus (eq. 6) entre $\overline{t_{ui}}$ et toutes les phrases t_{*i}, il est possible d'extraire celles les plus proches de la représentation généré.

$$cos(u, v) = \frac{u.v}{||u||.||v||} \in [-1, 1] \tag{6}$$

Un exemple de génération d'avis par extraction est donné Fig. 2. Bien qu'il soit impossible de tirer des conclusions d'un seul exemple, celui-ci montre clairement qu'on est capable d'extraire des phrases pertinentes.

8 Visualisation des caractéristiques apprises

Certains réseaux adverses permettent d'obtenir des espaces non-ambigus [CDH+16], où chaque dimension code pour une caractéristique particulière. Par exemple, [LZU+17] montre qu'il est possible d'ajouter des lunettes à quelqu'un en modifiant une composante latente du réseaux adverse. Attester de la non-ambiguïté d'un espace latent repose principalement sur l'observation des exemples générés. Dans notre cas, cette visualisation est moins évidente. En effet, nous générons un vecteur latent codant pour l'intégralité d'un avis et récupérons des mots/phrases existantes par similarité cosinus.

Pour essayer de visualiser l'effet d'une composante sur la génération, nous procédons de la sorte. Nous choisissons un couple (utilisateur,produit) au hasard et fixons toutes les composantes du vecteur "bruit" v à 0. A partir de cette entrée (u, i, v), nous générons un vecteur latent généré $\bar{t}_0$. Ce vecteur étant théoriquement dans le même espace que celui des mots, nous récupérons les n mots les plus proches de $\bar{t}_0$. Nous considérons cet ensemble $\{w_0\}$ comme notre base. A partir de

Vérité Terrain	Phrases les plus proches
I already played Bioshock.	Great sequel !
The game is more interesting.	This game has a great storyline !
The levels are bigger and more beautiful.	Great environment for an FPS.
There are new characters, new weapons.	For the most part this game is beautiful.
Enemies are clever and hard to kill .	Highly recommended for anyone that likes FPS games.
In conclusion this game is fantastic and beautiful.	I love the use of color and the art in general.

Table 2 – Exemple de prédiction par extraction à l'aide de la similarité cosinus entre le texte généré $\bar{t}$ et des phrases existantes.

là, nous répétons la procédure en modifiant une seule composante du vecteur de bruit, de 0 à 0.5 puis à 1 et de 0 à -0.5 puis à -1. A chaque fois, nous récupérons l'ensemble des mots les plus proches du vecteur généré : $\{w_{-0.5}\}\{w_{-1}\}\{w_{0.5}\}\{w_1\}$. Finalement, nous visualisons les différences entre notre ensemble base $\{w_0\}$ et tous les autres. En effet, les mots ajoutés sont liées au vecteur de bruit. La figure 4 représente tous ces mots pour la quatrième composante. Bien que dur à interpréter, changer la composante semble changer la polarité globale des mots.

9 Discussion

Ici, nous nous sommes intéressés à rendre le filtrage collaboratif plus explicite. En effet, pour être accepté, une suggestion se doit d'être accompagné d'une explication. Seulement, si l'explication est purement générée – et donc tout droit sortie d'une boite noire – elle nécessiterais elle-même une explication. En réalité, tout l'enjeu est d'obtenir un système suffisamment transparent et scrutable pour que l'utilisateur lui fasse confiance. Pour améliorer la transparence des systèmes de recommandations tout en restant dans le cadre classique du filtrage collaboratif, nous avons principalement proposé d'utiliser les réseaux adverses dans un cadre d'analyse de sentiment à l'aveugle.

Théoriquement, l'avantage d'un modèle adverse est qu'il peut combiner génération et espace latent non-ambigu. Ici, pour des raisons pratiques, nous utilisons des représentation pré-apprise et n'avons pas utilisé un décodeur directement textuel. De plus, l'architecture proposé, du fait de la personnalisation du générateur, ne permet probablement pas un apprentissage optimal de l'espace latent.Cependant, le modèle présenté ici, malgré ses défauts, fait figure de preuve de concept. Bien qu'incapable de générer directement du texte, il montre que l'exploitation de ces textes dans un cadre adverse permet notamment, en plus d'améliorer les performances, d'extraire du texte support – des mots, des phrases ou encore des avis entiers – pour aider l'utilisateur dans sa décisions.

Références

[AKCC15] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, 2015.

[BeK14] Shay Ben-elazar and Noam Koenigstein. A Hybrid Explanations Framework for Collaborative Filtering Recommender Systems. In *RecSys 2014*, 2014.

[BL07] James Bennett and Stan Lanning. The Netflix Prize. *KDD Cup and Workshop*, pages 3–6, 2007.

[BPL18] Homanga Bharadhwaj, Homin Park, and Brian Y Lim. Recgan : recurrent generative adversarial networks for recommendation systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 372–376. ACM, 2018.

[CC17] Rose Catherine and William Cohen. Transnets : Learning to transform for recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 288–296, New York, NY, USA, 2017. ACM.

[CDH+16] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan : Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

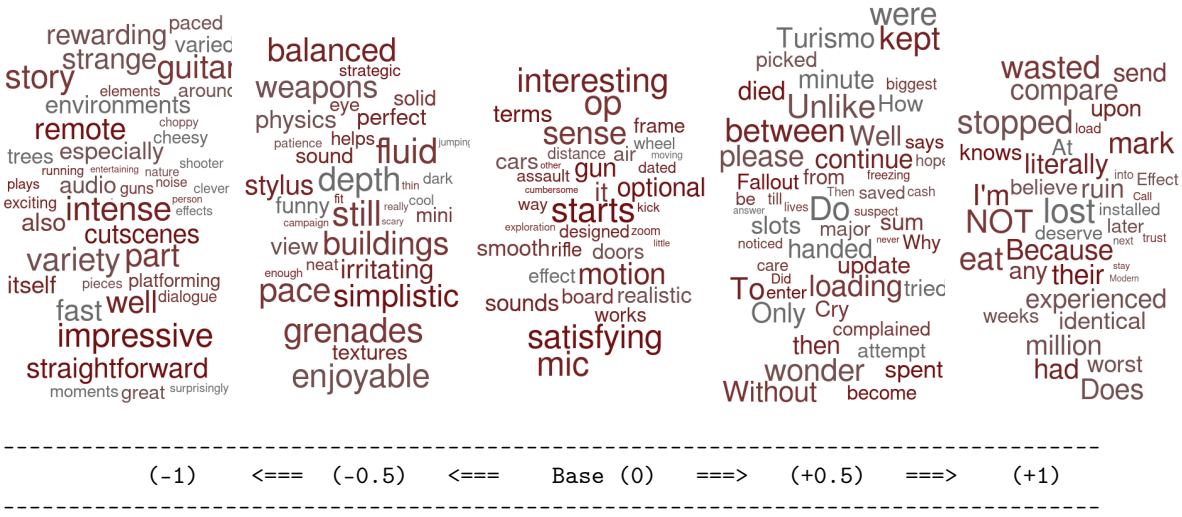


FIGURE 4 – Nuage de mots changeant en fonction d'une seule dimension du vecteur de bruit v

[CST+16] Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1650–1659, 2016.

[DdGGG18] Charles-Emmanuel Dias, Clara Gainon de Forsan de Gabriac, Vincent Guigue, and Patrick Gallinari. Rnn & modèle d'attention pour l'apprentissage de profils textuels personnalisés. *Proceedings of CORIA*, 2018.

[DGG16] Charles-Emmanuel Dias, Vincent Guigue, and Patrick Gallinari. Recommandation et analyse de sentiments dans un espace latent textuel. In *CORIA-CIFED*, pages 73–88, 2016.

[FGD18] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan : Better text generation via filling in the _. *arXiv preprint arXiv :1801.07736*, 2018.

[GLTFS87] Patrick Gallinari, Yann LeCun, Sylvie Thiria, and Francoise Fogelman-Soulie. Memoires associatives distribuees. *Proceedings of COGNITIVA*, 87 :93, 1987.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[HHDC18] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. Adversarial personalized ranking for recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 355–364. ACM, 2018.

[HKBT15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv :1511.06939*, 2015.

[HLZ+17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[KBV09] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42 :42–49, 2009.

[KLA18] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architec-

ture for generative adversarial networks. *arXiv preprint arXiv :1812.04948*, 2018.

[LZU+17] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, et al. Fader networks : Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, pages 5967–5976, 2017.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *Nips*, pages 1–9, 2013.

[ML13] J McAuley and J Leskovec. Hidden factors and hidden topics : understanding rating dimensions with review text. *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, pages 165–172, 2013.

[MPL15] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2015.

[NLVM17] Jianmo Ni, Zachary C Lipton, Sharad Vikram, and Julian McAuley. Estimating reactions and recommending products with generative models of reviews. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, volume 1, pages 783–791, 2017.

[SMH07] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[TM07] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. *Proceedings - International Conference on Data Engineering*, pages 801–810, 2007.

[ZNY17] Lei Zheng, Vahid Noroozi, and Philip S Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 425–434. ACM, 2017.

Détection automatique de structures blocs sur des matrices

Luce le Gorrec[1], Sandrine Mouysset[2], Daniel Ruiz[3], Philip A. Knight[4] et Iain S. Duff[5]

[4]University of Strathclyde, Glasgow
[5]Rutherford Appleton Laboratory, Didcot
[1,2,3]Université de Toulouse - IRIT, Toulouse

Résumé

Nous présentons un algorithme capable de partitionner les noeuds d'un graphe pondéré dirigé ou non, en détectant la structure par blocs de sa matrice d'adjacence. Grâce à un équilibrage doublement stochastique de cette matrice, l'intégralité de la structure par blocs est détectée à l'aide d'un minimum de vecteurs singuliers (en théorie un couple suffit).

Nous présentons les différentes étapes de notre algorithme, avant d'évaluer ses performances sur deux tâches de classification non supervisée : la détection de communautés et la détection de formes dans des nuages de points. En comparant les résultats obtenus à ceux d'autres algorithmes spécifiquement conçus pour ces tâches, nous montrons que notre algorithme est compétitif—dans le cas de la détection de communautés—, ou meilleur—dans le cas de la détection de formes—, par rapport aux méthodes existantes.

1 Introduction

Regrouper les éléments similaires dans un jeu de données sans information a priori est le but de la classification non supervisée (ou clustering) pour lequel de nombreuses techniques ont été proposées. Un des moyens couramment utilisés consiste à trouver des blocs dans les matrices représentant le jeu de données–telles que les matrices d'adjacence ou d'affinité, les tables de contingences ou Laplacien du graphe.

Le lien entre les propriétés spectrales et structurelles des matrices est un élément-clé de nombreux algorithmes de clustering [For10, Sch07, FMSV08]. Pour détecter des classes (ou clusters), les algorithmes spectraux classiques séparent en deux l'ensemble des noeuds de façon récursive, en suivant le signe des entrées d'un vecteur singulier. Cela peut être numériquement coûteux puisqu'il est nécessaire de calculer un nouveau vecteur singulier pour chaque bipartition. De plus, il se peut que la partition trouvée à chaque itération soit éloignée de la structure par blocs réelle de la matrice (un exemple est fourni Figure 3.1(c)). Les autres méthodes spectrales existantes [NJW01, VL07, LR+15] consistent à projeter les données dans le sous-espace vectoriel généré par les vecteurs propres dominants du Laplacien du graphe, en prenant autant de vecteurs qu'il y a de classes attendues, ce qui suppose que ce nombre est connu.

Dans ce papier, nous présentons une nouvelle approche spectrale pour le clustering. Notre but est de partitionner l'ensemble des noeuds d'un graphe pondéré, dirigé, et fortement connexe. Nous travaillons sur la matrice d'adjacence du graphe, notée $\mathbf{A}$. La principale nouveauté de notre méthode est de considérer les éléments singuliers de l'équilibrage doublement stochastique de la matrice $\mathbf{A}$. Trouver un équilibrage doublement stochastique pour une matrice $\mathbf{A}$ consiste à trouver deux matrices diagonales $\mathbf{D}$ et $\mathbf{F}$ telles que la somme sur chaque ligne et sur chaque colonne de $\mathbf{P} = \mathbf{DAF}$ est égale à 1. Nous montrons dans la Section 2 que l'équilibrage doublement stochastique améliore la fidélité de l'information structurelle contenue dans les vecteurs singuliers dominants de $\mathbf{P}$. En particulier, un faible nombre de vecteurs permet d'obtenir une information significative sur la structure par blocs sous-jacente de la matrice, le tout sans information complémentaire sur le nombre ou la taille des blocs.

En outre, travailler avec les vecteurs singuliers de $\mathbf{P}$ rend possible l'analyse des graphes dirigés et permet d'obtenir des clusterings précis, même lorsque la structure du graphe est très dissymétrique.

Le papier est construit de la manière suivante. En Section 2, nous exposons la connexion entre les vecteurs singuliers dominants d'une matrice doublement stochastique et sa structure par blocs. En Section 3 nous utilisons des outils du traitement du signal afin de détecter les informations sur le clustering contenues dans les vecteurs singuliers. Dans la Section 4, nous décrivons comment récupérer au fil des itérations des vec-

teurs singuliers qui continuent de fournir des informations utiles. La Section 5 est dédiée au réarrangement des blocs : nous devons compiler et analyser les clusterings venant des vecteurs singuliers gauche et droit, et éventuellement des itérations précédentes dans le cas où plus d'un couple de vecteurs est analysé. Nous présentons aussi une mesure adaptée à l'évaluation de la qualité de notre clustering et qui nous sert aussi pour notre critère d'arrêt. Enfin, des résultats empiriques sur des bancs d'essai sont présentés Section 6 afin d'indiquer l'efficacité de notre procédure.

2 Equilibrage doublement stochastique.

L'équilibrage doublement stochastique d'une matrice $\mathbf{A}$ existe à condition que $\mathbf{A}$ soit *bi-irréductible*, c'est-à-dire qu'il n'existe pas de permutations des lignes et des colonnes permettant de mettre $\mathbf{A}$ sous forme triangulaire par blocs. Comme cela est précisé Section 1, nous travaillons sur la matrice d'adjacence d'un graphe dirigé fortement connexe. $\mathbf{A}$ est donc carrée, positive, et bi-irréductible dès lors que sa diagonale ne comporte pas d'éléments nuls. Si nécessaire, nous ajoutons donc des termes non nuls dans sa diagonale, ce qui n'a pas d'effet sur la structure par blocs sous-jacentes que nous recherchons.

Le théorème suivant est une conséquence directe du théorème de Perron-Frobenius [Fro12, Per07].

Théorème 1 *Soit* $\mathbf{S} \in \mathbb{R}^{n\times n}$ *symétrique, irréductible et doublement stochastique. Alors* 1 *est valeur propre de* $\mathbf{S}$ *de multiplicité* 1*, et le vecteur propre associé est un multiple de* $\mathbf{e} = (1...1)^T$*. Par ailleurs, si* $\mathbf{S}$ *est symétriquement permutable en une matrice bloc diagonale avec* k *blocs irréductibles, alors* 1 *est valeur propre de multiplicité* k*, et une base du sous-espace propre associé est :*

$$\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\},$$

où v_{pq} *pour* $p \in \{1, \dots, k\}$ *est tel que*

$$v_{pq} = \begin{cases} 1, & \text{si } q \text{ est dans le bloc } p \\ 0, & \text{sinon.} \end{cases}$$

Ainsi, le calcul d'un vecteur propre associé à 1 d'une telle matrice $\mathbf{S}$ sera de la forme

$$\mathbf{x} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k.$$

En forçant $\mathbf{x} \in \mathbf{e}^\perp$, on peut raisonnablement supposer $a_i \neq a_j, \forall i \neq j$. Puisque ces vecteurs forment une partition disjointe de $\{1, \dots, n\}$, il est possible d'identifier la contribution de chaque bloc précisément, et donc de caractériser les partitions de manière exacte, en utilisant l'ensemble $\{a_1, \dots, a_k\}$. En effet, réordonner le vecteur singulier suivant un ordre adapté à la structure par blocs de $\mathbf{S}$ le mettra sous forme constante par morceaux. Cette idée est proche de celle exploitée dans [LN15], où les auteurs se servent de l'ordre croissant des vecteurs singuliers dominants de deux équilibrages stochastiques d'une matrice $\mathbf{A}$ afin de visualiser la structure par blocs de $\mathbf{A}$.

Comme corollaire du Théorème 1, considérons une matrice doublement stochastique non symétrique $\mathbf{P}$. Alors $\mathbf{P}\mathbf{P}^T$ et $\mathbf{P}^T\mathbf{P}$ sont toutes deux symétriques et doublement stochastiques, on peut donc leur appliquer le Théorème 1. Ainsi, si $\mathbf{P}\mathbf{P}^T$ ou $\mathbf{P}^T\mathbf{P}$ possède une structure par blocs, le vecteur singulier gauche ou droit dominant de $\mathbf{P}$ aura une contribution de chaque vecteur de la base associée, et il sera possible, comme précédemment, d'identifier la structure par blocs de lignes et/ou de colonnes de $\mathbf{P}$.

Notre algorithme est conçu pour des matrices qui sont des perturbations de matrices parfaitement diagonales par blocs. Les vecteurs singuliers dominants de telles matrices devraient avoir une structure proche d'une structure constante par morceaux. Ainsi, en réordonnant dans l'ordre croissant les vecteurs calculés, on devrait faire apparaître une structure proche d'une structure par blocs, ou au moins des blocs de lignes ou de colonnes faiblement corrélés les uns avec les autres. Le calcul de l'équilibrage doublement stochastique se fait avec la méthode de Newton décrite dans [KR13]. Cette méthode est peu coûteuse car elle ne fait intervenir que des produits matrices–vecteurs.

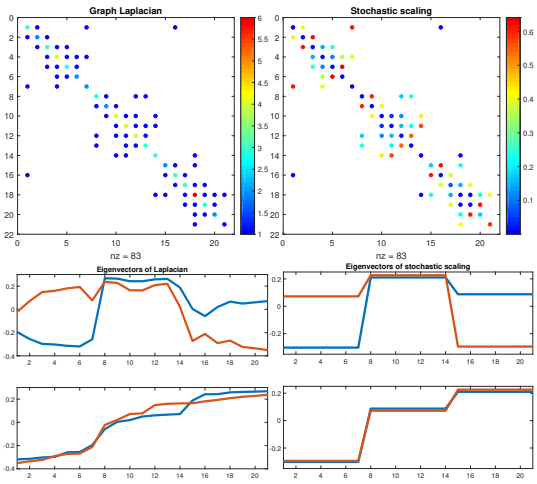


FIGURE 2.1 – Deux représentations matricielles d'un graphe et leurs vecteurs propres.

Afin de souligner l'avantage de l'équilibrage doublement stochastique par rapport à d'autres méthodes spectrales, nous utilisons le petit exemple Figure 2.1. Dans cet exemple, nous observons, pour un graphe présentant trois clusters distincts faiblement liés entre eux, l'information structurelle véhiculée par les vecteurs propres du Laplacien du graphe d'une part et ceux de sa matrice d'adjacence équilibrée d'autre part. Pour assurer la bi-irréducibilité, les termes diagonaux de la matrice d'adjacence sont fixés à 10^{-8} avant l'équilibrage. En bas de la Figure 2.1 on montre la structure numérique des vecteurs utilisés pour l'identification des clusters : les vecteurs associés aux deux plus petites valeurs propres pour le Laplacien (le vecteur de Fiedler est en bleu) ne permettent pas de détecter nettement les clusters (cela reste vrai pour le Laplacien normalisé). A contrario, les deuxième et troisième vecteurs propres dominants de la matrice stochastique permettent de caractériser les clusters parfaitement et sans ambiguïté.

3 Identification des clusters

Un point-clé de notre algorithme est d'identifier les clusters à l'aide des vecteurs singuliers de la matrice bi-irréductible doublement stochastique $\mathbf{P}$, sans connaissance préalable du nombre de clusters ni des permutations des lignes et des colonnes faisant apparaître la structure par blocs sous-jacente.

Les algorithmes de partitionnement qui travaillent sur des vecteurs analogues au vecteur de Fiedler divisent la matrice en deux blocs selon le signe des éléments du vecteur. Mais avec notre équilibrage doublement stochastique plus de deux blocs apparaissent sur un même vecteur, comme expliqué Section 2. Si $\mathbf{P}\mathbf{P}^T$ ou $\mathbf{P}^T\mathbf{P}$ a une structure sous-jacente quasi par blocs, ses vecteur propres dominants doivent avoir une structure quasi constante par morceaux après réordonnancement dans l'ordre croissant de ses entrées, comme dans la Figure 2.1. Notre problème est alors de détecter les paliers du vecteur réordonné que nous avons choisi d'envisager comme un problème de traitement du signal. Le vecteur peut en effet être considéré comme un signal 1D dont on cherche à détecter les sauts–ou les arêtes. Ainsi, nous procédons à la détection des clusters en appliquant des outils de traitement du signal, et notamment en effectuant un produit de convolution entre notre vecteur courant et un filtre spécifique. Les pics dans la convolution correspondent aux sauts dans le signal. Ces outils ont l'avantage de ne pas nécessiter la connaissance du nombre de clusters a priori.

Nous avons choisi d'utiliser le filtre de Canny [Can86], largement utilisé pour détecter des arêtes en traitement du signal et de l'image. Pour optimiser la détection des arêtes ce filtre combine trois critères : une bonne détection, une bonne localisation et la contrainte qu'une arête corresponde effectivement à un pic dans une convolution appropriée. Afin de satisfaire ces critères le filtre de Canny utilise un opérateur construit à partir du produit de convolution de la sortie d'un rapport signal/bruit (SNR) et d'une fonction de localisation (le filtre). La fonction de localisation optimale est la première dérivée du noyau Gaussien [Can86].

Nous avons adapté l'implémentation de la détection des arêtes afin de la rendre optimale pour notre application. Lors de l'utilisation du filtre, un paramètre nommé taille de la fenêtre glissante doit être fixé. Il détermine la raideur du filtre et l'étroitesse de son support. Comme montré dans la Figure 3.1, une faible valeur résulte en un filtre raide à support étroit qui va générer de nombreux pics dans la convolution, tandis qu'une grande valeur donne un filtre lisse à support large qui détectera moins de sauts.

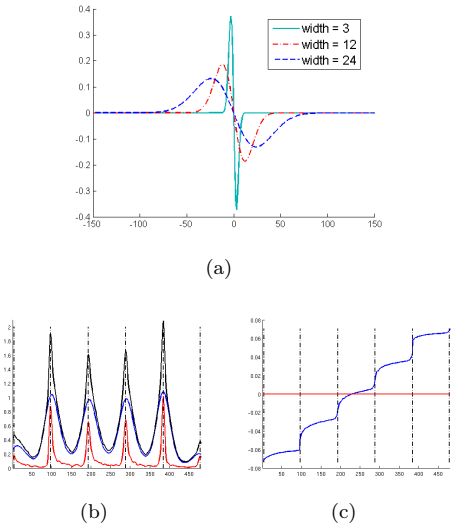


FIGURE 3.1 – (a) Le filtre pour différentes tailles de fenêtre glissante. (b) et (c) Détection des paliers.

Pour éviter les effets indésirables de la taille de la fenêtre, la détection des paliers est réalisée avec deux tailles de fenêtre différentes, une grande et une petite par rapport à la taille du vecteur. Nous travaillons ensuite sur la somme des convolutions afin de détecter les pics principaux et de s'assurer que nos sauts sont à la

fois nets et de taille suffisante.

La Figure 3.1(b) montre un exemple de l'effet de la fenêtre glissante sur la détection des sauts, via la matrice 480×480 `rbsb480` issue de la collection SuiteSparse [DH11]. Sur cette figure, il est clair que les sauts du vecteur sont mieux identifiés en utilisant la somme des convolutions. Dans la Figure 3.1(c), le vecteur singulier gauche est affiché afin d'indiquer la qualité des sauts détectés par les filtres. A contrario, la ligne rouge horizontale de la Figure 3.1(c) indique la bi-partition suggérée par le vecteur de Fiedler : en imposant d'avoir précisément deux clusters la séparation obtenue ne correspond finalement à aucun des paliers du vecteur.

4 SVD projetée

Si l'on suppose que $\mathbf{P}$ possède k blocs, alors les k vecteurs singuliers dominants devraient quasiment couvrir le sous-espace défini par les $\mathbf{v}_i$ décrits Section 2. En réalité, on peut trouver de nombreux blocs avec un unique couple de vecteurs singuliers. Mais on peut aussi itérer notre processus en utilisant une information complémentaire permettant d'affiner notre clustering. Chaque itération va nécessiter une nouvelle information spectrale, sinon nous redécouvrirons les mêmes blocs en boucle. Pour s'assurer que les nouveaux vecteurs apportent une information complémentaire nous travaillons avec les vecteurs propres dominants des équations normales de $\mathbf{P}$ projetées dans le sous-espace orthogonal à celui défini par les blocs déjà détectés. A la première itération, la projection s'effectue contre $Span(\mathbf{e})$. En effet, $\mathbf{e}$ est vecteur propre dominant de toute matrice doublement stochastique. Il sera renvoyé comme vecteur propre dominant de $\mathbf{P}\mathbf{P}^T$ et $\mathbf{P}^T\mathbf{P}$ si l'on ne prend pas garde à l'éviter, auquel cas une itération sera allouée à l'analyse d'un vecteur ne fournissant aucune information sur la structure par blocs de la matrice. Après la première itération, $\mathbf{e}$ appartient implicitement à l'espace des blocs identifiés.

La Figure 4.1 présente un exemple de ce procédé sur une matrice stochastique symétrique $\mathbf{P}$ ayant 4 blocs diagonaux. En utilisant le vecteur propre dominant de $\mathbf{P}$ orthogonal à $\mathbf{e}$ on détecte 3 blocs. Les deux blocs ayant les densités internes les plus faibles ne sont pas séparés (illustré Figure 4.1(b) avec le vecteur, dans son ordre naturel et dans l'ordre croissant de ses entrées). On projette ensuite $\mathbf{P}$ dans le complémentaire orthogonal de ces trois blocs. Le vecteur propre dominant de cette projection est affiché Figure 4.1(c). Il permet de séparer sans ambiguïté les blocs restés ensemble à l'itération précédente. Dans la Section 5 nous décrirons comment fusionner ces deux structures afin de découvrir la structure par blocs complète.

5 Amélioration des clusters

Nous expliquons maintenant comment combiner les différentes partitions en blocs de lignes ou de colonnes trouvées pour chaque vecteur analysé individuellement, ainsi que la superposition de la partition en blocs de lignes et celle en blocs de colonnes une fois le processus itératif terminé afin d'obtenir un clustering des noeuds du graphe. Nous présentons ensuite une mesure adaptée à l'évaluation de la qualité de nos partitions, qui permet aussi de fusionner certains petits blocs. Grâce à cette mesure, nous développons un critère d'arrêt pour déterminer la convergence du processus itératif.

Tout au long de cette section, nous illustrons nos propos avec la matrice `rbsb480` de la collection SuiteSparse [DH11] car elle possède une intéressante structure par blocs relativement fine sur ses lignes et ses colonnes. Nous remarquons que dans les Figures 5.1, 5.2 et 5.3, nous ne présentons pas des clusterings à proprement parler, mais des superpositions de partitions indépen-

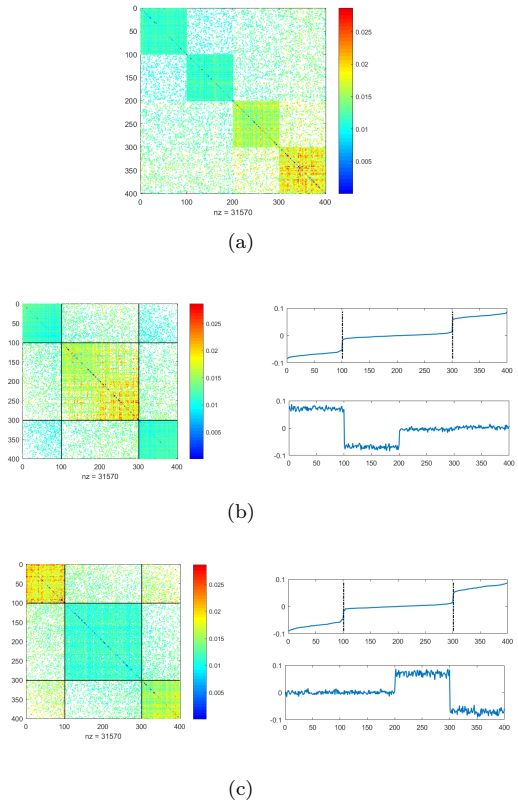


FIGURE 4.1 – Récupération d'une information additionnelle via les vecteurs propres.

dantes de lignes et de colonnes.

5.1 Superposition des partitions. Chaque fois que notre algorithme analyse un vecteur singulier droit (respectivement gauche) de la matrice, il trouve une partition des lignes (respectivement des colonnes). Evidemment, la partition peut être différente pour chacun des vecteurs analysés. Il est donc nécessaire d'incorporer la combinaison de ces partitions dans l'algorithme.

Ce processus est illustré Figure 5.1. La première étape de détection des blocs identifie 4 blocs de lignes et 5 blocs de colonnes, tandis que le deuxième couple de vecteurs analysés suggère une partition différente avec 3 blocs de lignes et 4 blocs de colonnes. Ces nouveaux blocs sont complémentaires des premiers et la combinaison des partitions résulte en 12 blocs de lignes et 20 blocs de colonnes, ces blocs étant bien séparés dans la matrice, comme le montre le schéma de gauche de la Figure 5.2.

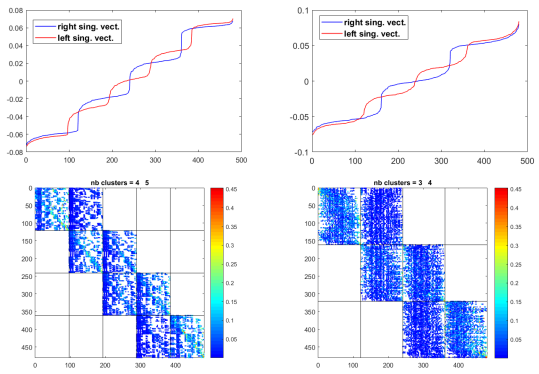


FIGURE 5.1 – Identification de blocs en utilisant différents vecteurs singuliers.

Nos tests sur `rbsb480` montrent l'intérêt d'analyser plusieurs vecteurs à la suite et de superposer les partitions en résultant. Cependant, ce processus peut aussi produire un partitionnement trop fine de la matrice initiale. Ce phénomène est illustré dans le schéma de droite de la Figure 5.2, où la fusion des partitions trouvées après trois processus d'analyse renvoie un découpage trop fin de la matrice (48 par 100 blocs). Cela motive le développement d'une méthode efficace pour fusionner de petits blocs.

5.2 Mesure de qualité. Nous basons l'analyse de nos blocs sur la mesure de modularité définie dans [New04]. La modularité de Newman peut être interprétée comme la somme sur toutes les classes de la différence entre la fraction de liens à l'intérieur de la

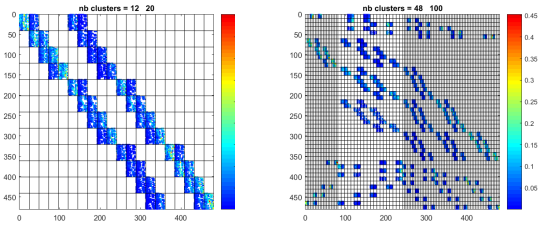


FIGURE 5.2 – Identification récursive des blocs obtenus en analysant plusieurs vecteurs à la suite.

classe et l'espérance de la fraction de ces liens dans un graphe aléatoire possédant la même distribution des degrés.

Dans le cas de notre matrice doublement stochastique $\mathbf{P}$, la partition des lignes peut être évaluée en utilisant la formule suivante :

$$\mathcal{Q}_r = \frac{1}{n}\sum_{k=1}^{r_C}\left(\mathbf{v}_k^T\mathbf{P}\mathbf{P}^T\mathbf{v}_k - \frac{1}{n}|\mathcal{J}_k|^2\right). \tag{5.1}$$

Cette équation est directement obtenue de la formulation de la modularité fournie dans [Bag12], en remarquant que $2m = n$ dans le cas doublement stochastique. Ici, n est le nombre de lignes de $\mathbf{P}$ et $|\mathcal{J}_k|$ est le nombre d'éléments du bloc k. De la même façon, on obtient une mesure de qualité $\mathcal{Q}_c$ pour la partition des colonnes en considérant la matrice $\mathbf{P}^T\mathbf{P}$, et en sommant sur l'ensemble des blocs de colonnes.

Il peut être démontré que notre mesure de qualité est comprise entre les bornes suivantes :

$$0 \leq \mathcal{Q}_r \leq 1 - \frac{1}{r_C},$$

et que la borne supérieure ne peut être atteinte que si les r_C blocs ont la même taille.

En dépit des limites bien connues de résolution de la modularité [FB07], cette mesure reste l'une des plus employées dans le domaine de la détection de communautés, et elle est adéquate pour répondre à notre problématique. En outre, dans le cas spécifique des graphes k-réguliers, plusieurs mesures incluant la modularité de Newman se comportent de la même façon [CC13]. Puisque les matrices doublement stochastiques peuvent être considérées comme les matrices d'adjacence de graphes pondérés 1-réguliers, les mesures de qualité communes donnent des résultats strictement équivalents.

Pour éviter l'apparition de mini blocs, nous fusionnons récursivement la paire de blocs dont la fusion augmente le plus la mesure de modularité, ce processus

s'arrêtant naturellement quand quelle que soit la paire de blocs, sa fusion fait décroître la valeur de la mesure.

Ce processus est très utilisé en détection de communautés, domaine dans lequel il est connu sous le nom d'algorithme glouton—voir par exemple [CNM04]—, mais il est généralement initié avec le clustering trivial dans lequel chaque classe contient un unique élément. Cette méthode est peu coûteuse numériquement et s'arrête sur un maximum local, ce qui peut être montré en adaptant la démonstration de [CNM04]. Le processus de fusion empêche ainsi une explosion du nombre de blocs, et il est appliqué après chaque superposition de partitions.

Le processus de fusion est illustré Figure 5.3. Le schéma de gauche montre la partition en 7 par 5 blocs obtenue après fusion de la partition en 12 par 20 blocs du schéma de gauche de la Figure 5.2. La mesure de qualité est alors de 0.5574 pour les lignes et 0.4932 pour les colonnes. Le schéma de droite montre la partition en 7 par 5 blocs obtenue après fusion de la partition en 48 par 100 blocs du schéma de droite de la Figure 5.2. Ici, les mesures de qualité valent 0.5574 pour les lignes et 0.502 pour les colonnes, et sont largement supérieures à celles obtenues pour la partition initiale.

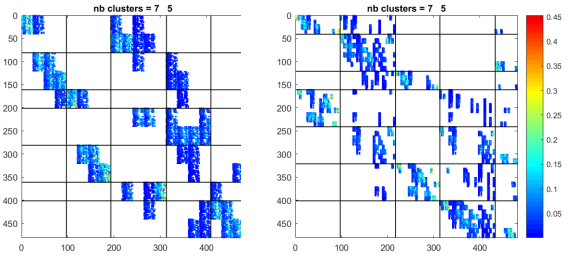


FIGURE 5.3 – Fusion des blocs paire à paire.

Pour arrêter le processus itératif, nous comparons la qualité de la partition mise à jour par le processus de fusion $\mathcal{Q}_r^{upd}$ avec celle de l'itération précédente $\mathcal{Q}_r^{ref}$. Quatre cas peuvent être rencontrés :

— Si $\mathcal{Q}_r^{upd} < \mathcal{Q}_r^{ref}$, on rejette le nouvel itéré.
— Si $\mathcal{Q}_r^{upd} > \mathcal{Q}_r^{ref}$ et le nombre de blocs est supérieur, on accepte l'itéré si le gain de modularité est significatif. Etant donnée la borne supérieure $\mathcal{Q}_r \leq 1 - \frac{1}{r_C}$, nous comparons le gain réel de modularité sur le gain théorique en utilisant le ratio

$$\rho = \frac{\mathcal{Q}_r^{upd} - \mathcal{Q}_r^{ref}}{\frac{1}{nbClust^{ref}} - \frac{1}{nbClust^{upd}}}.$$

Le seuil d'acceptation peut être ajusté pour contrôler le nombre d'itérations et la finesse de la partition. En pratique, une valeur du seuil entre 0.01 et 0.1 semble efficace.
— Si $\mathcal{Q}_r^{upd} > \mathcal{Q}_r^{ref}$ et le nombre de blocs n'est pas supérieur, on accepte l'itéré.
— Si les partitions sont identiques à une permutation près, la convergence est atteinte.

6 Applications

Nous illustrons les performances de notre algorithme sur la détection de communautés, et le testons ensuite sur des jeux de données pour le clustering issus de la librairie scikit-learn [PVG+11].

6.1 Détection de communautés. Nous testons d'abord notre algorithme sur des réseaux simples. Pour assurer la bi-irréducibilité avec un effet minimal sur la structure, nous forçons les valeurs diagonales initialement nulles à 10^{-8}.

Comme prétraitement, nous supprimons les entrées dominantes (> 0.55) de la matrice après équilibrage, en considérant qu'il s'agit de blocs préalablement détectés pendant le processus décrit dans la Section 4. Sinon, ces entrées impliquent des vecteurs propres dominants quasiment canoniques dont les sauts seront difficiles à détecter. Puisque ces éléments ne sont pas réellement des communautés—il s'agit de hubs ou de noeuds pendants—, nous réincorporons ces noeuds aux communautés en utilisant notre processus de fusion des blocs une fois la convergence de l'algorithme atteinte.

Nous avons comparé notre algorithme (US) avec quatre algorithmes de détection de communautés reconnus, à savoir Walktrap (WT) [PL05], Louvain (ML) [BGLL08], Fast and Greedy (FG) [CNM04], et Leading Eigenvector (LE) [New06], en utilisant l'implémentation de la librairie `igraph` [CN06]. WT et ML sont conçus pour travailler directement sur la structure du graphe, tandis que LE se concentre sur l'exploitation de l'information spectrale. FG est l'algorithme glouton mentionné dans la Section 5.2.

Nous avons appliqué ces algorithmes sur quatre ensembles de réseaux de 500 noeuds, générés avec le banc d'essai de Lancichinetti–Fortunato–Radicchi (LFR) [LFR08], qui sont largement utilisés dans le domaine de la détection de communautés pour ses caractéristiques proches de celles des réseaux issus du monde réel. Chaque ensemble de réseaux correspond à un degré moyen différent $\overline{k}$. Les détails sur la génération des graphes sont donnés Table 6.1. Nous appelons "exp. db des degrés" l'exposant de la loi de puissance utilisée pour générer la distribution des degrés, et "exp. db des tailles de communautés" l'exposant de la loi de puissance utilisée pour générer la distribution des tailles

Table 6.1 – Paramètres pour générer les réseaux.

Paramètre	Valeur
nombre de noeuds	500
degré moyen $\overline{k}$	{10,20,40,75}
degré maximum	$2 \times \overline{k}$
exp. db des degrés	-2
exp. db des tailles de communautés	-1

des communautés.

Nous testons le comportement des algorithmes en utilisant le paramètre de mélange qui quantifie la force d'une structure en communautés. A l'origine, ce paramètre mesure la force d'appartenance d'un noeud à une communauté via le ratio entre ses liens à l'extérieur de la communauté et son degré. Plus ce paramètre est grand pour chaque noeud, plus la structure est faible. Le paramètre de mélange du réseau μ est la moyenne des paramètres de mélanges nodaux.

Nous mesurons la précision de la structure trouvée par les algorithmes en utilisant l'information mutuelle normalisée (NMI). Cette mesure vient de la théorie de l'information (voir par exemple [FJ03]). Elle mesure l'écart entre deux partitions candidates qui n'ont pas nécessairement le même nombre de classes. La NMI est très utilisée en détection de communautés [YAT16]. Elle est à valeurs dans $[0, 1]$, valant 1 si les deux partitions sont identiques.

Les résultats sont donnés Figure 6.1. Chaque schéma correspond à la valeur de $\overline{k}$ précisée au-dessus. Pour générer ces courbes, nous avons généré 50 réseaux pour chaque valeur de μ puis calculé la NMI moyenne. Un point d'une courbe correspond donc à un couple (μ,NMI moyenne). Nous pouvons diviser les algorithmes existants en deux groupes : ML et WT, dont la NMI est proche de 1 même pour de grandes valeurs de μ, et FG et LE dont la NMI décroît rapidement. Notre algorithme est entre ces deux groupes.

Nous notons que les comportements de ces deux groupes se rapprochent quand $\overline{k}$ augmente. Ceci peut être expliqué par les contraintes imposées lors de la génération des réseaux qui vont fournir des communautés plus grosses quand $\overline{k}$ est grand, comme montré Figure 6.2. Ici, les matrices d'adjacence de deux réseaux de nos jeux de données sont affichées. Dans ces réseaux, $\mu = 0.3$. Dans le réseau de gauche, $\overline{k} = 10$ avec 40 communautés ; dans celui de droite $\overline{k} = 75$ avec 5 communautés. Les grosses communautés sont généralement mieux détectées par les algorithmes. Cependant, ce facteur n'est pas le seul déterminant pour notre algorithme car sa courbe de NMI est plus sensible à l'accroissement de $\overline{k}$ que FG et LE.

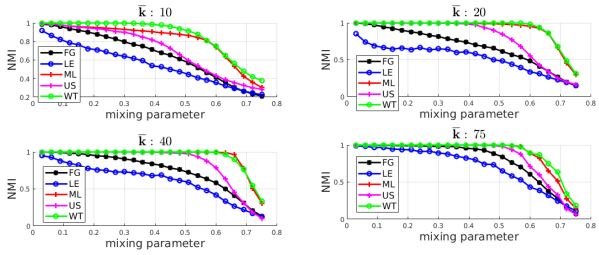


Figure 6.1 – Courbes de NMI pour FG, LE, ML, WT, US.

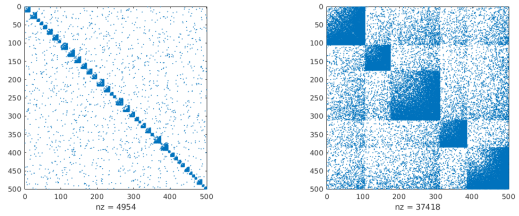


Figure 6.2 – Matrices d'adjacence de réseaux.

Nous avons souhaité savoir si ce comportement pouvait être lié au fait que les vecteurs propres dominants de réseaux très creux véhiculent parfois d'autres informations que la structure en communautés [FH16]. Nous avons donc comparé notre algorithme initial (USd) à notre algorithme appliqué sur le réseau perturbé (USw), c'est-à-dire sur $\mathbf{A} + \epsilon \mathbf{e}\mathbf{e}^T$ avec $\mathbf{A}$ la matrice d'adjacence du réseau. Nous avons fixé ϵ à 0.15 comme cela est suggéré pour l'algorithme Pagerank de Google [BP98]. Ces deux versions sont comparées Figure 6.3 pour deux valeurs de $\overline{k}$ différentes. On observe que les deux versions ont un comportement similaire quand $\overline{k} = 0.75$, tandis que USw (courbe bleue) est bien plus précis que USd (courbe rose) pour $\overline{k} = 20$.

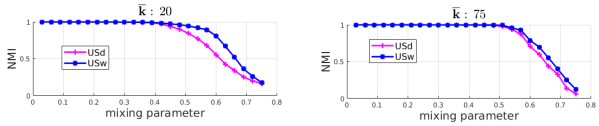


Figure 6.3 – Courbes de NMI pour USd et USw.

Pour compléter cette étude, nous avons observé le

comportement de WT, ML, USd et USw pour des valeurs croissantes de $\overline{k}$, comme montré Figure 6.4. ML, WT et USw commencent par accroître leur précision, semblent atteindre un seuil, puis voient leur performance décroître à mesure que $\overline{k}$ augmente. USd augmente d'abord aussi sa précision, mais ses performances ne décroissent pas pour la valeur maximale de $\overline{k}$.

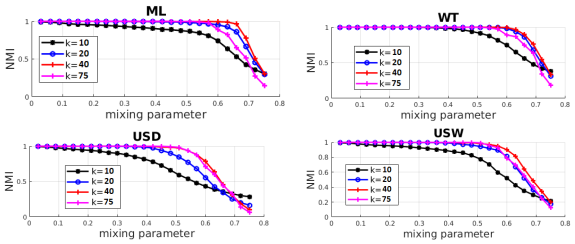


FIGURE 6.4 – Courbes de NMI de ML, WT, USd, USw suivant des valeurs de $\overline{k}$.

Ainsi, bien que notre algorithme ne soit pas nécessairement meilleur que tous les algorithmes de détection communautés existant sur des réseaux très creux, il est capable de mieux détecter les communautés que des algorithmes conçus à cet effet et largement utilisés. En outre, il semble être une alternative très encourageante quand les réseaux deviennent plus denses, puisque ses performances ne décroissent pas dans ces circonstances. Nous verrons d'ailleurs Section 6.2 qu'il obtient d'excellents résultats pour détecter les structures par blocs de matrices d'affinités, qui peuvent être envisagées comme les matrices d'adjacence de graphes pondérés complets.

Enfin, notre algorithme n'est pas contraint à travailler avec des matrices symétriques et fournit donc une méthode polyvalente pour la détection de communautés dans les graphes dirigés, même lorsque ces graphes présentent des flots déséquilibrés—i.e. un déséquilibre entre les arêtes entrantes et sortantes d'une communauté—, alors que WT et ML—tout deux contraints de travailler sur $\mathbf{A} + \mathbf{A}^T$—renvoient des résultats erronés dans ce cas. Un exemple est montré Figure 6.5, où deux communautés sont fortement connectées de façon dissymétrique. Les résultats de ML, WT et USd sont fournis, les lignes noires représentant les séparations entre les communautés. Seul notre algorithme renvoie ici un résultat cohérent.

6.2 Jeux de données pour clustering. Afin de montrer les capacités de notre algorithme sur d'autres tâches de clustering, nous l'avons testé sur les jeux de données de la librairie scikit-learn [PVG+11]. Il

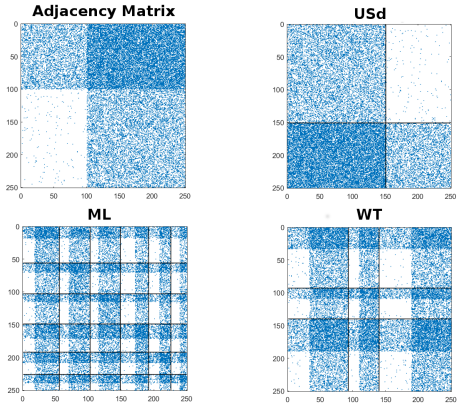


FIGURE 6.5 – Détection de classes dissymétriques.

s'agit pour l'algorithme de détecter des formes cohérentes dans des nuages de 1500 points en 2D à partir de la matrice d'affinité du nuage. Afin de montrer la modulabilité de notre méthode, nous avons mis l'accent sur les différents types de formes à détecter : certains de ces nuages peuvent être séparés linéairement, d'autres non et l'un d'eux est fortement anisotrope. La matrice d'affinité [NJW01] d'un ensemble de points $\{x_i \in \mathbb{R}^p, i = 1...n\}$ est la matrice symétrique définie par

$$\mathbf{A}_{i,j} = \begin{cases} \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2}), & \text{pour } i \neq j, \\ 0, & \text{sinon.} \end{cases} \tag{6.1}$$

La précision de notre méthode va fortement dépendre du choix du paramètre Gaussien σ. Nous le fixons en suivant l'heuristique de [MNR08] pour prendre en compte à la fois la densité et les dimensions du nuage.

Puisque la matrice d'affinité est à diagonale nulle, nous forçons la bi-irréducibilité en fixant les éléments diagonaux à 10^{-8}. Comme en Section 6.1 nous supprimons les entrées dominantes de la matrice en guise de pré-traitement. Comme post-traitement, chacune de ces entrées est ensuite ajoutée à son plus proche cluster, c'est-à-dire celui contenant le point qui lui est le plus proche au sens de la distance Euclidienne.

Nous avons comparé notre méthode avec celles proposées dans la librairie scikit-learn. Les résultats pour cinq de ces méthodes sont fournies Figure 6.6. Chaque ligne correspond à un jeu de données spécifique, chaque colonne à une méthode précisée ci-dessus. Deux points d'une même couleur ont été assignés au même cluster. La colonne de droite montre les résultats de notre méthode, que nous avons arrêtée après l'analyse d'un

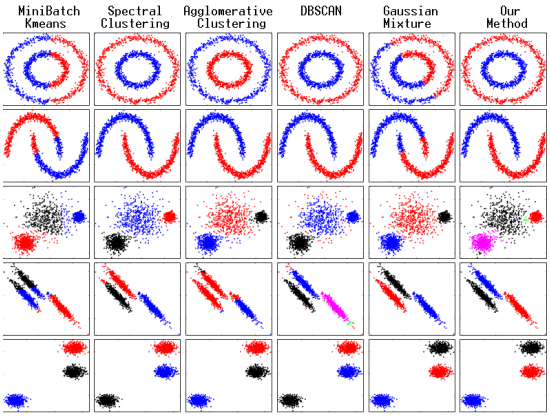


FIGURE 6.6 – Comparaison d'algorithmes de clustering.

TABLE 6.2 – NMI des clusterings Figure 6.6.

M.B. Kmeans	Spect. Clust.	Agglo. Clust.	DB SCAN	Gauss. Mixt.	Our method
$2.9.e^{-4}$	1	0.993	1	$1.3.e^{-6}$	1
0.39	1	1	1	0.401	1
0.813	0.915	0.898	0.664	0.942	0.902
0.608	0.942	0.534	0.955	1	0.996
1	1	1	1	1	1

unique vecteur propre. Etant donnés ces résultats, il est clair que ce vecteur fournit une information suffisante pour séparer grossièrement les nuages en clusters cohérents pour tous les jeux de données quelles que soient leurs formes. La NMI est donnée Table 6.2, en suivant le même ordre ligne/colonne que la Figure 6.6. On voit qu'excepté pour deux jeux de données—le quatrième où un élément a été mal assigné pendant le post-traitement, et le troisième pour lequel la NMI est supérieure à 0.9— notre méthode a toujours le meilleur score. Nous remarquons aussi qu'à part `DBSCAN` (4eme colonne), toutes ces méthodes nécessitent de connaître à l'avance le nombre de classes.

7 Conclusions

Nous avons développé un algorithme spectral capable de partitionner une matrice avec peu de vecteurs singuliers, principalement grâce aux propriétés de l'équilibrage doublement stochastique. Notre méthode fonctionne en outre sans connaissance a priori du nombre de classes, et ne nécessite pas de "symétriser" la matrice artificiellement.

Nous l'avons ensuite utilisée sur des problèmes standards d'analyse de données, où nous l'avons confronté avec des méthodes spécifiquement conçues pour ces problèmes. Notre algorithme s'est avéré compétitif, mais il est en plus polyvalent, car conçu pour être appliqué à des matrices quel que soit ce qu'elles représentent. Pour changer d'application, il suffit donc d'adapter le pré- et le post-traitement.

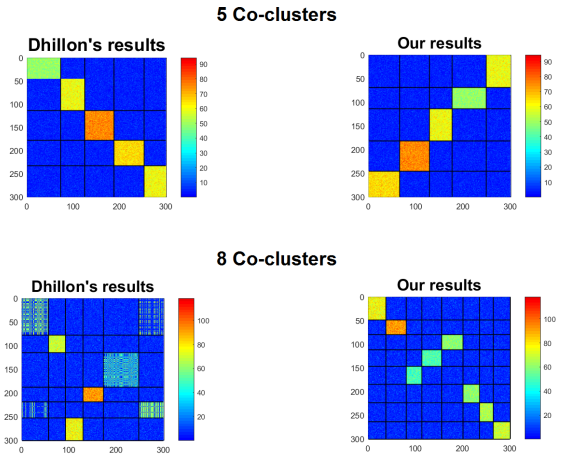


FIGURE 7.1 – Détection de co-clusters.

Le but de nos travaux futurs sera de généraliser notre méthode pour effectuer des tâches de co-clustering. En effet, elle est capable de détecter des structures rectangulaires dans les matrices carrées, comme le montre la Figure 7.1. Ici, deux matrices présentant une structure à blocs rectangulaires ont été générées avec la fonction `make_biclusters` de scikit-learn [PVG+11]. A droite, nous voyons que notre algorithme est capable de détecter sans erreur ces blocs rectangulaires dans les deux matrices, tandis que l'algorithme de Dhillon [Dhi01]–à gauche–échoue à détecter certains des blocs de la matrice du bas.

Ce court exemple fournit des perspectives encourageantes pour le co-clustering. Cependant, les résultats de la Figure 7.1 ne montrent que la superposition de partitions indépendantes sur les lignes et les colonnes, sans exploiter les liens entre ces partitions, alors qu'il est connu ([DMM03, LN15]) que l'imbrication entre les clusters lignes et colonnes est un paramètre important de la qualité d'un co-clustering.

Une étude sur la complexité est en cours, notamment au niveau du passage à l'échelle de l'algorithme. En attendant, une implémentation simple de l'algorithme en Matlab peut être téléchargée sur : *http ://cloud.irit.fr/index.php/s/2S6hZzO16R4JPJI.*

Références

[Bag12] James P Bagrow. Communities and bottlenecks : Trees and treelike networks have high modularity. *Physical Review E*, 85(6) :066118, 2012.

[BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 2008(10) :P10008, 2008.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7) :107–117, April 1998.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8 :679–698, 1986.

[CC13] Patricia Conde-Cespedes. Modelisation et extension du formalisme de l'analyse relationnelle mathematique a la modularisation des grands graphes. *PhD Thesis, Universite Pierre et Marie Curie*, 2013.

[CN06] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems :1695, 2006.

[CNM04] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(6) :066111, December 2004.

[DH11] Timothy A Davis and Yifan Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1) :1–14, 2011.

[Dhi01] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 269–274, New York, NY, USA, 2001. ACM.

[DMM03] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic co-clustering. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 89–98, New York, NY, USA, 2003. ACM.

[FB07] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1) :36–41, 2007.

[FH16] Santo Fortunato and Darko Hric. Community detection in networks : A user guide. *CoRR*, abs/1608.00163, 2016.

[FJ03] Ana L. N. Fred and Anil K. Jain. Robust data clustering. In *CVPR (2)*, pages 128–136. IEEE Computer Society, 2003.

[FMSV08] David Fritzsche, Volker Mehrmann, Daniel B. Szyld, and Elena Virnik. An SVD approach to identifying metastable states of Markov chains. *Electronic Transactions on Numerical Analysis*, 29 :46–69, 2008.

[For10] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3) :75–174, 2010.

[Fro12] Georg Frobenius. Ueber matrizen aus nicht negativen elementen. *Sitzungsber. Königl. Preuss. Akad. Wiss*, page 456–477, 1912.

[KR13] Philip A. Knight and Daniel Ruiz. A fast algorithm for matrix balancing. *IMA Journal of Numerical Analysis*, 33(3) :1029–1047, 2013.

[LFR08] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4) :046110, October 2008.

[LN15] L. Labiod and M. Nadif. A unified framework for data visualization and coclustering. *IEEE Transactions on Neural Networks and Learning Systems*, 26(9) :2194–2199, Sep. 2015.

[LR+15] Jing Lei, Alessandro Rinaldo, et al. Consistency of spectral clustering in stochastic block models. *The Annals of Statistics*, 43(1) :215–237, 2015.

[MNR08] Sandrine Mouysset, Joseph Noailles, and Daniel Ruiz. Using a global parameter for gaussian affinity matrices in spectral clustering. In *VECPAR*, volume 5336 of *Lecture Notes in Computer Science*, pages 378–390. Springer, 2008.

[New04] M.E.J. Newman. Analysis of weighted networks. *Physical Review E*, 70 :056131, 2004.

[New06] Mark E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3) :036104, 2006.

[NJW01] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering : Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems : Natural and Synthetic*, NIPS'01, pages 849–856, Cambridge, MA, USA, 2001. MIT Press.

[Per07] Oskar Perron. Zur theorie der matrices. *Mathematische Annalen*, 64(2) :248–263, 1907.

[PL05] P. Pons and M. Latapy. Computing communities in large networks using random walks (long version). *ArXiv Physics e-prints*, December 2005.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011.

[Sch07] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1) :27–64, 2007.

[VL07] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4) :395–416, 2007.

[YAT16] Zhao Yang, René Algesheimer, and Claudio J. Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, 6 :30750 EP –, Aug 2016. Article.

AutoCV Challenge Design and Baseline Results

Liu Zhengying
U. Paris-Sud / U. Paris-Saclay
Inria Saclay
Orsay, France
zhengying.liu@inria.fr

Isabelle Guyon
U. Paris-Sud / U. Paris-Saclay
Inria Saclay
Orsay, France
guyon@clopinet.com

Julio C. S. Jacques Junior
U. Oberta de Catalunya / Computer Vision Center
Barcelona, Spain
jsilveira@uoc.edu

Meysam Madadi
Computer Vision Center
Barcelona, Spain
mmadadi@cvc.uab.es

Sergio Escalera
U. of Barcelona / Computer Vision Center
Barcelona, Spain
sergio@maia.ub.es

Adrien Pavao
Inria Saclay
Orsay, France
adrien.pavao@gmail.com

Hugo Jair Escalante
INAOE-CINVESTAV, Mexico
ChaLearn USA
hugo.jair@gmail.com

Wei-Wei Tu
4Paradigm Inc.
Beijing, China
tuwwcn@gmail.com

Zhen Xu
Ecole Polytechnique
Palaiseau, France
zhen.xu@polytechnique.edu

Sébastien Treguer
La Paillasse
Paris, France
streguer@gmail.com

Abstract

We present the design and beta tests of a new machine learning challenge called AutoCV (for Automated Computer Vision), which is the first event in a series of challenges we are planning on the theme of Automated Deep Learning[1]*. We target applications for which Deep Learning methods have had great success in the past few years, with the aim of pushing the state of the art in fully automated methods to design the architecture of neural networks and train them without any human intervention. The tasks are restricted to multi-label image classification problems, from domains including medical, areal, people, object, and handwriting imaging. Thus the type of images will vary a lot in scales, textures, and structure. Raw data are provided (no features extracted), but all datasets are formatted in a uniform tensor manner (although images may have fixed or variable sizes within a dataset). The participants's code will be blind tested on a challenge platform in a controlled manner, with restrictions on training and test time and memory limitations. The challenge is part of the official selection of IJCNN 2019.*

[1] https://autodl.chalearn.org/

1. Introduction

Machine learning, and deep learning in particular, has achieved considerable success in recent years. This in a wide variety of tasks ranging from computer vision, to natural language processing. State-of-the-art in deep learning, aligned with the increasing computational power of personal computers and accessible Graphic Processing Units (GPU) with strong computational capabilities, caused a revolution with respect to traditional computer vision, with unprecedented and outstanding results in different types of tasks such as image classification, recognition, and cap-

tioning, among others. However, so far this success crucially relies on human intervention in many steps (e.g., for data pre-processing, feature engineering, model selection, hyperparameter optimization, etc.). As the complexity of these tasks is often beyond non-experts, the rapid growth of deep learning applications has created a demand for off-the-shelf or reusable methods, which can be used easily and without expert knowledge. It is in this context that the Automated Deep Learning (AutoDL) field arises.

Previous and pioneer work on Automated Machine Learning (AutoML) in the context of supervised learning (see, e.g., [8]) comprises the basis for taking autonomy into the context of deep learning (AutoDL). In broad terms, the goal of AutoML is to develop "universal learning machines" capable of solving supervised learning problems without any user intervention. Great progress has been achieved in this topic, with quite competitive solutions (e.g., [5]) being publicly available to the average user. Such solutions, however, have been designed to work on tabular formatted data. That is, the feature extraction and representation strategy is responsibility of the user. AutoDL goes one step further, by attempting to automate the feature extraction and representation processes, in addition to the learning step. Ideally, with AutoDL we would like to find an autonomous solution that receives as input any type of input data and automatically generates a model that solves the associated task. As a first step in such direction, we organize the AutoCV challenge in which we ask participants to develop automatic methods for learning from raw visual information.

The objective of the new AutoCV challenge is to address some of the limitations of the previous challenges and to provide an ambitious benchmark for code wrappers around TensorFlow [1], which should be able to solve multi-label classification problems without any human intervention, in limited time, on any large-scale dataset. Data will be formatted in a uniform way as series of example-label pairs having identical corresponding structures, e.g. $\{X_1, Y_1\}$, $\{X_2, Y_2\}$, etc. This will lend itself in particular to the use of convolutional neural networks (CNN), where X_i are images, Y_i are binary vectors indicating which classes X_i belongs to. Although the use of Tensorflow will be encouraged by providing participants with a starting kit including sample code demonstrating how to solve the problems at hand, the participants will be free to provide solutions with any deep learning / machine learning framework, such as scikit learn [18] and PyTorch [17].

This paper describes the design of the AutoCV challenge, the first competition of its kind that aims at automating model construction for the analysis of visual information. A distinctive feature of this challenge is that models will be able to deal with raw data directly, hence making it a perfect testbed for representation learning and CNN based methods. Please note that although we expect deep learning models to obtain the best performance, participants can propose solutions based in any machine learning formulation. This competition is the first of a series that aims at boosting research on AutoDL.

The article is organized as follows. In Section 1.1, we provide a mathematical formulation of the central problem: the AutoML problem. In Section 1.2, we summarize related work. In Section 2, we present the overall competition design with details on the competition protocol, data, metrics used for evaluation, etc. In Section 3, we introduce several baseline methods and report their performance on formatted datasets. We conclude our paper in section 4.

1.1. Mathematical Formulation of the Problem

Since the AutoCV challenge aims to tackle the AutoML problem within the field of computer vision, we provide herein a mathematical formulation of the AutoML problem. We focus on the supervised learning case for current challenge.

Suppose we have a dataset $D = \{(x_i, y_i)\}_{i=1}^n$, assocaited to a supervidsed learning task, where $x_i \in \mathcal{X}$ are examples (e.g. images) and $y_i \in \mathcal{Y}$ are their corresponding labels (e.g. "dog" or "cat" or a binary vector). To define a *task*, we first do a *train/test split* $D = D_{tr} \cup D_{te}$ then separate examples and labels in test set to get $D_{te}^{\emptyset} = \{x|(x, y) \in D_{te}\}$ and $Y_{te} = \{y|(x, y) \in D_{te}\}$. A supervised learning task (either classification or regression) is then defined by the 5-tuple [2]

$$\mathcal{T} = (D_{tr}, D_{te}^{\emptyset}, L, B_T, B_S)$$

where $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a loss function measuring the performance $L(y', y)$ of predictions y' with respect to true labels y, B_T and B_S are time and space budget restrictions, respectively (these budgets are given with respect to a fixed universal Turing machine, e.g. a modern computer). The goal of the task is to find an *algorithm*

$$f : x \mapsto y$$

capable of mapping each example x to a prediction y of its corresponding label, within time and space budget limits B_T and B_S. Where we want this algorithm f to be as "good" as possible at making predictions. Such goodness is measured by a performance function $P(f; D_{te}, L)$ typically defined by

$$P(f; D_{te}, L) = -\frac{1}{|D_{te}|} \sum_{(x,y) \in D_{te}} L(f(x), y)$$

Then the problem can be formulated as (**Hard-coding Problem**)

$$\arg\max_f P(f; D_{te}, L). \tag{1}$$

[2] In some scenarios, the set of test examples $D_{te}^{\emptyset}$ is unknown at training time too and we have for theses cases $\mathcal{T} = (D_{tr}, L, B_T, B_S)$.

An important remark about (1) is that the test labels in D_{te} are *unknown* at training time. This also implies that the values of the objective function in (1) are unknown too. The final performance is only known at test time. Thus to find a satisfying solution for (1), some estimation techniques are often applied, using either human intuition or statistical methods (e.g. using a validation set).

To solve (1) automatically, machine learning approaches exploit the dataset D_{tr} in the task $\mathcal{T} = (D_{tr}, D_{te}^{\emptyset}, L, B_T, B_S)$. The key is to construct a *learning algorithm*

$$A : \mathcal{T} \mapsto f$$

that takes a task $\mathcal{T}$ as input and outputs an algorithm f, within time budget B_T and space budget B_S. The problem then becomes (**Machine Learning Problem**)

$$\arg\max_{A} P(\hat{f}; D_{te}, L), \text{ where } \hat{f} = A(\mathcal{T}). \tag{2}$$

Note that the maximization task of finding A in (2) is usually done by human experts, or more specifically by *machine learning experts*.

In real applications and machine learning research, the learning algorithm A is usually encoded by many hyperparameters (e.g. kernel of SVM, neural network architecture, optimizer parameters such as learning rate, regularizer parameters such as weight decay, etc), which turn out to be crucial to obtain acceptable performance. So to stress out the importance of these hyperparameters, sometimes we also write learning algorithms as

$$A_\lambda, \lambda \in \Lambda$$

with λ representing the hyperparameters of A_λ and Λ the hyperparameter space. Manually tuning these hyperparameters is a time consuming trial-and-error process, and this is where AutoML enters. A big part of AutoML's goal is indeed to automate the hyperparameter tuning process. This subdomain of AutoML is called *hyperparameter optimization* (HPO). An *HPO algorithm*

$$\mathfrak{A} : \mathcal{T} \mapsto A$$

aims to *automatically* solve (2), i.e. to automatically find a learning algorithm A given a task $\mathcal{T}$. A typical process of an HPO algorithm goes like: try different learning algorithms A_j (with different hyperparameters) on $\mathcal{T}$, estimate their performance P_j and in the end suggest the most promising one(s). In the literature, the HPO problem is sometimes defined as the Full Model Selection (FMS) problem [4] or the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem [22]. We recall the CASH problem as follows. Given training set D_{tr}, we do a k-fold cross-validation [12] by splitting D_{tr} into k equal-sized partitions $D_{tr} = D_{va}^{(1)} \cup \cdots \cup D_{va}^{(k)}$ and write $D_{tr}^{(i)} = D_{tr} \backslash D_{va}^{(i)}$. Consider a set of algorithms $\mathrm{A} = \{A^{(1)}, \ldots, A^{(l)}\}$ with associated hyperparameter spaces $\Lambda^{(1)}, \ldots, \Lambda^{(l)}$. The CASH problem is defined by

$$\arg\max_{A^{(j)} \in \mathrm{A}, \lambda \in \Lambda^{(j)}} \frac{1}{k} \sum_{i=1}^{k} P(f_{ij}; D_{va}^{(i)}, L), \text{ where } f_{ij} = A_\lambda^{(j)}(D_{tr}^{(i)}).$$

We see that this CASH formulation actually tries to solve (2) automatically by using k-fold cross-validation as estimation of the test score $P(f; D_{te}, L)$. Again, as in the case of classic machine learning formulation, this CASH formulation for AutoML is but one possible solution to the real problem (2), with manual choice of A, $\Lambda^{(i)}$, cross validation as estimation, fold number k, etc.

In real life, machine learning experts never deal with only one single task $\mathcal{T}$ in their life. They gain experience by working on many different tasks $\{\mathcal{T}_j\}$ and often become more and more efficient at finding good learning algorithms. More formally, this process can be described as follows. The past experience is described by[3]

$$\mathfrak{D} = \{(\mathcal{T}_j, A_j, P_j)\}_{j=1}^{N}$$

with P_j being (an estimation of) the performance got by applying learning algorithm A_j to task $\mathcal{T}_j$. And we wish to construct a *meta-learning* algorithm

$$\mathcal{A} : \mathfrak{D} \mapsto \mathfrak{A}$$

that *learns* an HPO algorithm $\mathfrak{A}$ by exploiting past experience $\mathfrak{D}$. Note that the set of past experience $\mathfrak{D}$ can be dynamically enriched during the process of applying $\mathcal{A}$, since $\mathcal{A}$ can typically try a new learning algorithm A_{new} on a task $\mathcal{T}$ and get a performance P_{new}. Then we can append the 3-tuple $(\mathcal{T}, A_{new}, P_{new})$ to $\mathfrak{D}$ since it becomes past experience too. We note here the similarity between this process and what we have in *reinforcement learning*. Keeping the dynamic nature of $\mathfrak{D}$ in mind, we can consider hyperparameter optimization as a special case of meta-learning with $\mathfrak{D} = \emptyset$ at beginning and that the search of A is done only on a fixed task $\mathcal{T}$. Although in the literature, the term "meta-learning" lays more emphasis on the fact of using past experience on *different* datasets to a new dataset. So to avoid confusion, we'll call both an HPO algorithm or a meta-learning algorithm an AutoML algorithm.

But what is the objective of $\mathcal{A}$? Given a list $\mathfrak{D}_{te}$ of tasks to solve (recall that $\mathfrak{D}_{te}$ contains 5-tuples of the form $\mathcal{T} := (D_{tr}, D_{te}, L, B_T, B_S)$), one form of the objective can be formulated as (**AutoML Problem**)

$$\boxed{\begin{array}{c} \arg\max_{\mathcal{A}} \sum_{\mathcal{T} \in \mathfrak{D}_{te}} P(\hat{f}; D_{te}, L) \\ \text{s.t. } \hat{f} = \hat{A}(\mathcal{T}) \text{ and } \hat{A} = \hat{\mathfrak{A}}(\mathcal{T}), \hat{\mathfrak{A}} = \mathcal{A}(\mathfrak{D}). \end{array}} \tag{3}$$

[3] We can also consider $\mathfrak{D} = \{\mathcal{T}_j, A_j, f_j, P_j\}_{j=1}^{N}$ with trained $f_j = A_j(\mathcal{T}_j)$. Then we are talking about *transfer learning* instead of meta-learning. But here we'll make no distinction for simplicity.

From this definition, we see that an AutoML algorithm $\mathcal{A}$ should exploit $\mathfrak{D}$ (a dataset of tasks and past experience) and produce $\mathfrak{A}$ (an HPO algorithm), which will produce (or select) a learning algorithm $\hat{A}$ for each task $\mathcal{T}$, and the goal is to maximize the sum (or equivalently the average) of the performance on all tasks to solve in $\mathfrak{D}_{te}$. To add more clarity, we'll call $\hat{\mathfrak{A}} = \mathcal{A}(\mathfrak{D})$ the *meta-learning step*, $\hat{A} = \hat{\mathfrak{A}}(\mathcal{T})$ the *hyperparameter optimization step* and $\hat{f} = \hat{A}(\mathcal{T})$ the usual *supervised learning step*. And here we emphasize the importance of the imposed time and space budget limits B_T and B_S for each task. This restriction is crucial for an AutoML algorithm to be applicable in real-life scenarios.

In addition to the definition of the AutoML problem in (3), another important aspect that AutoML intends to explore is the *any-time learning* framework for AutoML algorithms. In an any-time learning setting, an AutoML algorithm $\mathcal{A}$ should produce (for each task $\mathcal{T}$) a ready-to-predict learning algorithm A_t (or simply f_t) at any timestamp $t \in \mathbb{R}_+$. And we judge the performance of $\mathcal{A}$ based on the sequence $\{A_t\}_{t\in\mathbb{R}_+}$ instead of considering just the final one A_T. This means that we want our AutoML algorithm to not only produce correct predictions at final time but also be fast and efficient. And this urges AutoML algorithms to carefully deal with exploration-exploitation trade-off. It's not hard to image that this any-time learning setting of the challenge will make participants' algorithms more robust and ready to be applied to real-life problem solving.

The problem approached in the AutoCV challenge is that of solving the AutoML problem defined in (3) for scene understanding tasks in an any-time learning setting.

1.2. State of the Art

According to Murdock et al. [16], while much manual engineering effort has been dedicated to the task of designing deep architectures that are able to effectively generalize from available training data, model selection is typically performed using subjective heuristics by experienced practitioners. Although freeing users from the troublesome handcrafted feature extraction by providing a uniform feature extraction classification framework, DCNNs still require a handcrafted design of their architectures [15]. Hence, most deep architectures for image classification learn shared image representations with a single model.

It is well known that lower layers of deep neural networks tend to represent low-level image features while higher layers encode higher-level concepts. Intuitively, and as reported in [16], categories that are more similar should share more information than those that are very different. Thus, training independent fine-grained models for these subsets of related labels could result in specialized features that are tuned to differentiating between subtle visual differences between similar categories.

Hierarchical deep networks [23] attempt to incorporate information from a known hierarchy to improve prediction performance without requiring architecture changes by learning multi-task models with shared lower layers and parallel, domain-specific higher layers. However, hyperparameters such as the location of branches and the relative allocation of nodes between them must still be specified prior to training. Ideally, model selection should be performed automatically, allowing the architecture to adapt to training data. As a step towards this goal, authors [16] proposed *Blockout*, an approach for simultaneously learning both the model architecture and parameters. Inspired by *Dropout* [10], Blockout can be viewed as a technique for stochastic regularization that adheres to hierarchically-structured model architectures.

Pérez-Rúa et al. [19] addressed the problem of finding an optimal neural architecture design for a given image classication task. They proposed the *Efficient Progressive Neural Architecture Search* (EPNAS) by aggregating two state-of-the-art in neural architecture search, i.e., *Sequential Model-Based Optimization* (SMBO) [13], and increasing training efficiency by sharing weights among sampled architectures [20]. Carreira-Perpinán and Idelbayev [3] addressed the task of pruning neural networks, which consists of removing weights without degrading its performance. In their work, pruning is formulated as an optimization problem of finding the weights that minimize the loss while satisfying a pruning cost condition. Nevertheless, the method neither has strong influence on the model architecture nor on the model selection. Ma and Xia [15] proposed a genetic DCNN designer to automatically generate the architecture of a DCNN for each given image classification problem. First, they partition a DCNN into multiple stacked meta convolutional blocks and fully connected blocks. Then, they use refined evolutionary operations, including selection, mutation and crossover to evolve a population of DCNN architectures. The main limitation of this approach, as mentioned by the authors, is that although the training of each generated DCNN is limited to 100 epochs, the proposed genetic DCNN designer has an extremely high computational and space complexity, due to storing and evaluating a large number of DCNN structures.

According to Cai et al. [2] techniques for automatically designing deep neural network architectures such as reinforcement learning based approaches have recently shown promising results. However, their success is based on vast computational resources, making them difficult to be widely used. A noticeable limitation is that they still design and train each network from scratch during the exploration of the architecture space, which is highly inefficient. In their work, a new framework toward efficient architecture search is proposed by exploring the architecture space based on the current network and reusing its weights. A reinforcement learning agent is employed as the meta-controller, whose

action is to grow the network depth or layer width with function-preserving transformations. Thus, previously validated networks can be reused, optimizing computational cost. The method was applied to explore the architecture space of the plain convolutional neural networks (no skip-connections, branching etc.).

As it can be seen, automated computer vision (i.e., model architecture design and selection, hyper-parameter setting, etc.) become an emerging research line, with a wide range of possibilities (i.e., with respect to applications and ways to be exploited), in particular when deep learning is employed. Different approaches have been proposed in the past few years, without a general solution. Furthermore, the difficulty of comparing existing approaches relies on the lack of benchmarks. This way, we expect the design of the present AutoCV Challenge can help to advance the state-of-the-art on this field.

2. Competition Design

This section describes the design of the AutoCV challenge. As previously mentioned, this competition asks participants to develop AutoML methods to solve scene understanding tasks and starting from raw data.

2.1. Competition Protocol

AutoCV challenge adopts the same competition protocol as the upcoming AutoDL challenge [14]. The evaluation process, for a single task, is shown in Figure 1. In this competition, the same submission will be evaluated on 5 different tasks and the evaluation is handled in parallel, in an asynchronous/parallel manner.

Some major differences between the AutoCV challenge and prior AutoML challenges [6, 7] are:

1. **Raw data:** Data are no longer pre-processed in a uniform feature vector representation; instead, they are formatted into TFRecords (a standard generic data format used by TensorFlow [1]) by keeping their raw nature. For datasets with images under standard compression formats (e.g. JPEG, BMP, GIF), we directly use the bytes as data and decode them on the fly to have a 3D tensor for each image.

2. **Large scale datasets:** For development, datasets will all be under 4GB (after compression), for practical reasons (See Section 2.2), however, for final testing, datasets of hundreds of thousands of examples will be used.

3. **Any-time learning:** The metric of evaluation (based on learning curves, see Section 2.3) will force the participants to provide algorithms that can make good predictions as early as possible.

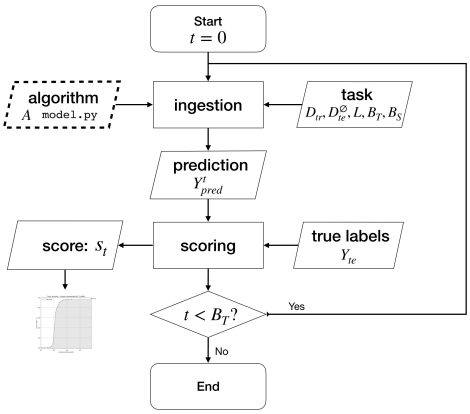


Figure 1: **AutoCV challenge's evaluation process** for *one* task defined by $D_{tr}, D_{te}^{\emptyset}, L, B_T, B_S$. Challenge participants have to provide a Python script named `model.py` implementing the logic of their AutoCV algorithm (shown as the dotted parallelogram). To evaluate its performance, this script is imported in ingestion program (defined in `ingestion.py`) in which it is trained on D_{tr} and produces a prediction Y_{pred}^{t} on $D_{te}^{\emptyset}$, where t is the timestamp. The prediction Y_{pred}^{t} is then compared to true labels Y_{te} in scoring program (defined in `score.py`) and produces a score s_t. This ingestion/scoring process is repeated until the time budget T is used up (or until `model.py` actively stops further training). At any time, the score sequence $s_{t_0}, s_{t_1}, \ldots$ is visualized as a learning curve and the area under learning curve is used as the evaluation for this task. Note that scoring is running in parallel with ingestion thus the time for computing scores is not counted in the time participants' code consumed (but the time used for training and predicting is).

4. **Fixed resource learning:** The participants will be given limited memory and computational resources to run their code. They will be informed of resources made available to them (number of cores, memory, GPU type, etc.). Their (compressed) code size will be limited to 300 MB, to allow submission of pre-trained models with moderate complexity. Thus, **pre-training** will be allowed, provided that the submission size limit is respected.

5. **Uniform tasks and metrics:** All problems will be multi-label classification problems and will be evaluated with the same metric (see Section 2.3).

One key aspect of this challenge, and other past AutoML challenges [8] we organized, is that it is a **code submission** challenge. Participants will submit code that will be trained

and tested on the challenge platform (see Sec. 2.5) without any human intervention, on datasets they will never see.

The challenge will be run in one single phase. Feedback score will be immediately provided on a leaderboard, using 5 **feedback datasets** (invisible to the participants, but visible to their submitted code). And this score is used for final evaluation.

In the challenge, we have 2 types of datasets: **public** and **feedback**. In contrast with some previous AutoML challenges [6, 7] in which the feedback data was distributed to the participants (except for the target values of the validation and test sets), in the AutoCV challenge, the practice datasets won't be exposed to the participants directly. Their code will be fully blind tested. However, several fully labeled "public" datasets will be provided, and we intend to set up a **public repository** to encourage participants to exchange among themselves other datasets and to enable meta-learning. A **starting kit** in Python with a TensorFlow interface will be provided (Sec. 2.4). Moreover, several examples of sample submissions will be provided.

2.2. Data

We provide 10 image datasets for the AutoCV challenge. They come from 5 different domains as follows: 1) **handwriting** recognition; 2) **object** recognition; 3) **People** related images (faces/emotions, etc); 4) **Aerial** images, and 5) **Medical** images. For either the 5 public datasets or the 5 feedback datasets, one dataset from each of the 5 domains is used. A detailed description about the 5 public datasets that are available to download is shown in Table 1.

All tasks are supervised learning problems, i.e., data samples are provided in pairs $\{X, Y\}$, X being an input image/tensor (which may have different number of channels for each domain) and Y a target binary vector. The problem is slightly simplified by the fact that for all images within a particular dataset, the bundle structure will be fixed.

2.3. Evaluation Metrics

AutoCV challenge enforces *any-time learning* by scoring participants with the Area under the Learning Curve (ALC) (Figure 2).

The participants can train in increments of a chosen duration (not necessarily fixed) to progressively improve performance, until the time limit is attained. Several predictions can be made during the learning process and this allows us to plot their learning curves, i.e., "performance" as a function of time. More precisely, for each prediction made at a timestamp, we compute for each (binary) class the *Area Under ROC Curve* (AUC), then normalize it (and average over all classes) by

$$NAUC = 2 * AUC - 1.$$

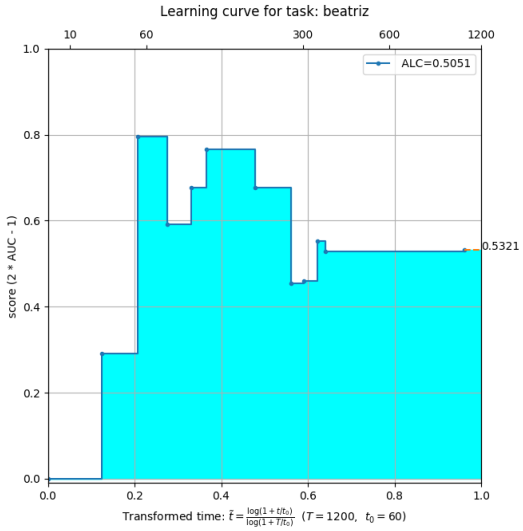


Figure 2: **Example of learning curve.** We modified the CodaLab competition platform (Sec. 2.5) so participants can save their results, at any intervals they choose, to incrementally improve their performance, until the time limit is attained. In this way, we can plot their learning curves: performance as a function of time. By evaluating them with the area under the learning curve, we push them to implement any-time learning methods. The x-axis corresponds to timestamp but normalized to [0,1]. This figure shows an example of possible over-fitting in which the participant could have stopped further training earlier.

Then for each dataset, we compute the Area under Learning Curve (ALC) of a submission as follows:

- at each timestamp t, we compute $s(t)$, the $NAUC$ (see above) of the most recent prediction. In this way, $s(t)$ is a step function with respect to timestamp t;

- in order to normalize time to the $[0, 1]$ interval, we perform a time transformation by

$$\tilde{t}(t) = \frac{\log(1 + t/t_0)}{\log(1 + T/t_0)}$$

  where T is the time budget (e.g. 1200 seconds = 20 minutes) and t_0 is a reference time amount (e.g. 60 seconds).

- then compute the area under learning curve using the

Table 1: **AutoCV datasets summary.** We provide 10 datasets for the AutoCV challenge, among which 5 **public** datasets are shown. Similar datasets have been formatted for the feedback and test phases, but will remain hidden for the participants. "row" and "col" are image size (height and width, respectively). "var" indicates the dimension is not fixed and can vary across examples.

#	Dataset	Domain	Class number	Sample number		Tensor dimension		
				train	test	row	col	channel
1	Munster	hand-writing	10	60000	10000	28	28	1
2	Chucky	objects	100	48061	11939	32	32	3
3	Pedro	people	26	80095	19905	var	var	3
4	Decal	aerial	11	634	166	var	var	3
5	Hammer	medical	7	8050	1965	600	450	3

formula

$$\begin{aligned} ALC &= \int_0^1 s(t)d\tilde{t}(t) \\ &= \int_0^T s(t)\tilde{t}'(t)dt \\ &= \frac{1}{\log(1+T/t_0)} \int_0^T \frac{s(t)}{t+t_0} dt \end{aligned}$$

we see that $s(t)$ is weighted by $1/(t+t_0)$, giving a stronger importance to predictions made at the beginning of th learning curve.

The ALC gives the evaluation score for one task. Finally, when ALC score is computed for all tasks, the final score is obtained by the **average rank** (over all tasks among all submissions). It should be emphasized that multi-class classification metrics are not being considered, i.e., each class is scored independently.

2.4. Starting Kit

To encourage participants to participate in the AutoCV Challenge, a starting kit, which can be downloaded from the competition web page, will be provided. The purpose of this starting kit is two-fold:

- **Submission file**: participants can directly upload this zip file as a submission. They can also deploy their own method and prepare a ZIP file for submission;
- **Local test environment**: it allows participants to have a complete local submission handling environment (for a single dataset). Thus they can easily run local tests, using their own method (implemented in model.py) for different dataset.

Both above utilities can be done following instructions in the `README.md` file, contained in the starting kit[4].

[4] https://github.com/zhengying-liu/autodl_starting_kit_stable

2.5. CodaLab: platform for running competitions

As in previous versions of AutoML Challenge, the AutoCV challenge will run on CodaLab. The CodaLab platform[5] is a powerful open source framework for running competitions that involve result or code submission. For AutoCV Challenge, we created our own CodaLab instance[6] so that computational resources can be adapted to the needs of the Challenge. CodaLab facilitates the organization of computational competitions from the organizers point of view, as well as provide a rich set of tools (e.g., forum, leaderboard, online submission, etc) to help participants. In AutoCV, participants will be able to submit their codes, and receive (almost) real-time feedback on a leaderboard (i.e., during a development phase), where the performances of different participants can be compared (as illustrated in Fig. 3).

3. Baseline Methods and Experimental Results

3.1. Baseline Methods

In this section, we present baseline methods with different model complexity and requiring different amount of computing resources, which have been applied on the datasets used in the AutoCV Challenge.

3.1.1 Linear Classifier with Basic Scheduling

This baseline method uses a very simple architecture: a neural network with no hidden layer. In the feed-forward phase, the input tensor is flattened then fully connected to the output layer, with a sigmoid activation. During training, it uses a sigmoid cross entropy loss (i.e., as if we are doing several binary logistic regression independently at the same time) and Adam optimizer [11] with default learning rate. The batch size is fixed to 30 for both training and test.

If the input shape is variable, some preprocessing procedure is required. When it is the case, we simply resize

[5] https://competitions.codalab.org/
[6] https://autodl.lri.fr

#	User	Entries	Date of Last Entry	<Rank> ▲	Dataset 1 ▲	Dataset 2 ▲	Dataset 3 ▲	Dataset 4 ▲	Dataset 5 ▲	Detailed Results
1	Zhengying	6	04/04/19	3.0000	0.0000(3)	0.0000(3)	0.0000(3)	0.0000(3)	0.0000(3)	Learning Curve
2	jaihoj	4	03/19/19	2.0000	0.1796(2)	0.0459(2)	0.0000(2)	0.0110(2)	0.0001(2)	Learning Curve
3	Pavao	9	03/17/19	1.0000	0.5839(1)	0.1899(1)	0.0007(1)	0.0386(1)	0.0002(1)	Learning Curve

Figure 3: Illustration of the leaderboard shown in our CodaLab instance, developed specifically for AutoCV.

all images to have fixed shape 112×112 (the number of channels is always fixed).

As we are in an any-time learning setting in AutoCV challenge, we need to have a working predictor at any time. According to the design of the challenge (Figure 1), this means that we also need a strategy for scheduling training and test. In this method, we propose following scheduling strategy. At the beginning, the algorithm trains the neural network for $s = 10$ steps. An estimation of time used per step is computed. Then the number of training steps gets doubled ($s \leftarrow 2s$) at each train/test call and the algorithm computes the duration required for this number of training steps using the estimation. This estimated duration is then compared to remaining time budget (sent by ingestion program). If there is still enough time, continue another call of training; otherwise, actively stop the evaluation process.

3.1.2 Convolutional Neural Networks Trained from Scratch

In this second baseline method, we use ResNet50 V2 [9]. The input image is resized to 128×128 and normalized with respect to the mean and standard deviation of all image pixel values. We assign sigmoid activation to the last layer for all types of classification tasks, i.e. multi-class and multi-label. The network is initialized with Xavier-Glorot initialization and trained with cross entropy loss and Adam optimizer with learning rate 0.001. We do not use any data augmentation during training. The data is shuffled in each epoch. The scheduling strategy is the same as the one defined in Sec. 3.1.1.

3.1.3 Neural Networks with Pre-trained Weights

In our third baseline method, we use pre-trained Inception V3 [21]. The Inception family appeals to many tricks in order to improve the image classification performance. All image inputs are resized to be 299×299. For grey images, we convert it to 3 channel and for images more than 3 channels, we extract the RGB channel. We use pre-trained weight provided by Tensorflow and fine-tune the model with cross entropy loss and default Adam optimizer. No data augmentation nor advanced techniques are used. Data is shuffled every epoch. The scheduling strategy is the same as that in section 3.1.1.

3.2. Experiments on Public and Feedback Datasets

We ran above baseline methods on the formatted datasets. All these experiments are carried out on Google Cloud virtual machine instances under Ubuntu 18.04, with one single GPU (Nvidia Tesla P100) and 16 GB Memory. The time budget is fixed to 20 minutes for all tasks.

The results are presented in Table 2. We show results on public and feedback datasets.

From Table 2, we see that the difficulty of different tasks vary a lot. Indeed, the difficulty depends on many parameters such as image shape, number of examples, number of classes, etc. This imposes participants' algorithm to be as flexible and robust as possible to provide satisfying any-time solution.

We note that in some cases, even though the final performance of ResNet/pre-trained Inception is better than that of linear baseline (e.g. in the case of Munster or Saturn), its overall ALC score is not better than linear baseline. This can be explained by the fact that the training is longer for ResNet (when the number of training steps is fixed) but the performance is not significantly better at the starting point. Thus it gains less Area under Learning Curve at beginning.

4. Conclusion and Further Work

We presented the design of a new challenge to stimulate the AutoML community to embrace deep learning and tackle the hard problems of automating architecture design and hyper-parameter search for models trained directly on raw data. We have run baseline methods and verified the feasibility of the tasks, given the allotted time and computational resources. The results will be reported at IJCNN 2019 and feedback from the community will be sought.

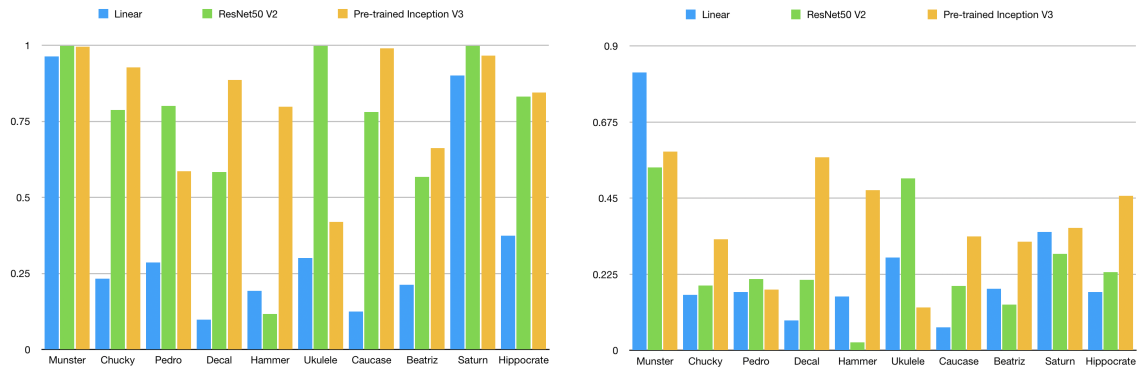
5. Acknowledgements

This project receives support from Google Zurich, ChaLearn and 4Paradigm Inc. This work has been partially supported by the Spanish project TIN2016-74946-P (MINECO/FEDER, UE) and CERCA Programme / Generalitat de Catalunya. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPU used for this research. This work is partially supported by ICREA under the ICREA Academia programme. We would like to thank our numerous collaborators for

Table 2: **Baseline results** on public and feedback datasets used in AutoCV challenge. Three baseline methods (linear, ResNet and pre-trained Inception) are applied to all 10 datasets. Performances are in final normalized AUC (NAUC) and Area under Learning Curve (ALC). These results are further visualized in Figure 4a and Figure 4b.

	#	Dataset	Domain	Linear		ResNet50 V2 [9]		Pre-trained Inception V3 [21]	
				NAUC	ALC	NAUC	ALC	NAUC	ALC
Public	1	Munster	hand-writing	0.9628	0.8223	0.9999	0.5408	0.9950	0.5883
	2	Chucky	objects	0.2331	0.1643	0.7877	0.1914	0.9270	0.3289
	3	Pedro	people	0.2863	0.1733	0.8009	0.2115	0.5867	0.1805
	4	Decal	aerial	0.0982	0.0893	0.5833	0.2085	0.8861	0.5712
	5	Hammer	medical	0.1922	0.1596	0.1173	0.0238	0.7986	0.4742
Feedback	1	Ukulele	hand-writing	0.3003	0.2747	0.9986	0.5093	0.4189	0.1276
	2	Caucase	objects	0.1249	0.0683	0.7801	0.1910	0.9897	0.3367
	3	Beatriz	people	0.2129	0.1829	0.5675	0.1350	0.6621	0.3212
	4	Saturn	aerial	0.9003	0.3507	0.9987	0.2860	0.9665	0.3621
	5	Hippocrate	medical	0.3743	0.1726	0.8314	0.2319	0.8452	0.4571

Figure 4: **Visualization of baseline results.**



(a) Normalized AUC (NAUC) of the final predictions of the three baseline methods, after 20 minutes of training.

(b) Area under Learning Curve (ALC) of the three baseline methods, with 20 minutes of training.

their help and support. We thank André Elisseeff, Olivier Bousquet, Mahsa Behzadi, Tatiana Merkulova for helping challenge design and early development. Some of our datasets are collected and donated by Mehreen Saeed, Jun Wan, Stephane Ayache, Alexandre Gramfort, Hubert Jacob Banville, Sébastien Tréguer. We thank Lisheng Sun, Herilalaina Rakotoarison, Michèle Sebag, Marc Schoenauer, Kristin Bennett, Sébastien Tréguer, Danny Silver, Baiyu Chen, Shangeth Rajaa, Gavin Cawley, Albert Clapés i Sintes, Jun Wan, Quanming Yao, Mengshuo Wang, Yi-Qi Hu, Ju Xu, Jingsong Wang for participating in beta test and/or making useful suggestions. The CodaLab platform backend and web development are supported by Tyler Thomas and Eric Carmichael.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 2, 5

[2] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang. Efficient architecture search by network transformation. The Thirty-Second AAAI Conferenceon Artificial Intelligence, 2018. 4

[3] M. A. Carreira-Perpinán and Y. Idelbayev. learning-compression algorithms for neural net pruning. In Proceedings of the IEEE Conference on Computer Vision

and Pattern Recognition, pages 8532–8541, 2018. 4

[4] H. J. Escalante, M. Montes, and L. E. Sucar. Particle Swarm Model Selection. Journal of Machine Learning Research, 10(Feb):405–440, 2009. 3

[5] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 2962–2970. Curran Associates, Inc., 2015. 2

[6] I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, Tin Kam Ho, N. Macia, B. Ray, M. Saeed, A. Statnikov, and E. Viegas. Design of the 2015 ChaLearn AutoML challenge. pages 1–8. IEEE, July 2015. 5, 6

[7] I. Guyon, I. Chaabane, H. J. Escalante, S. Escalera, D. Jajetic, J. R. Lloyd, N. Maci, B. Ray, L. Romaszko, M. Sebag, A. Statnikov, S. Treguer, and E. Viegas. A Brief Review of the ChaLearn AutoML Challenge: Any-time Any-dataset Learning Without Human Intervention. In Workshop on Automatic Machine Learning, pages 21–30, Dec. 2016. 5, 6

[8] I. Guyon, L. Sun-Hosoya, M. Boullé, H. Escalante, S. Escalera, Z. Liu, D. Jajetic, B. Ray, M. Saeed, M. Sebag, et al. Analysis of the automl challenge series 2015-2018, 2017. 2, 5

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In European conference on computer vision, pages 630–645. Springer, 2016. 8, 9

[10] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. CoRR, abs/1207.0580, 2012. 4

[11] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs], Dec. 2014. arXiv: 1412.6980. 7

[12] R. Kohavi. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. event-place: Montreal, Quebec, Canada. 3

[13] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In The European Conference on Computer Vision (ECCV), September 2018. 4

[14] Z. Liu, O. Bousquet, A. Elisseeff, S. Escalera, I. Guyon, J. Jacques, A. Pavao, D. Silver, L. Sun-Hosoya, S. Treguer, W.-W. Tu, J. Wang, and Q. Yao. AutoDL Challenge Design and Beta Tests-Towards automatic deep learning. In MetaLearn workshop @ NIPS2018, Montreal, Canada, Dec. 2018. 5

[15] B. Ma and Y. Xia. Autonomous deep learning: A genetic DCNN designer for image classification. CoRR, abs/1807.00284, 2018. 4

[16] C. Murdock, Z. Li, H. Zhou, and T. Duerig. Blockout: Dynamic model selection for hierarchical deep networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2583–2591, June 2016. 4

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 2

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and . Duchesnay. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12(Oct):2825–2830, 2011. 2

[19] J.-M. Pérez-Rúa, M. Baccouche, and S. Pateux. Efficient progressive neural architecture search. arXiv preprint arXiv:1808.00391, 2018. 4

[20] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In J. Dy and A. Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 4095–4104, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. 4

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2818–2826, Las Vegas, NV, USA, June 2016. IEEE. 8, 9

[22] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. arXiv:1208.3719 [cs], Aug. 2012. arXiv: 1208.3719. 3

[23] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu. Hd-cnn: Hierarchical deep convolutional neural networks for large scale visual recognition. In The IEEE International Conference on Computer Vision (ICCV), pages 2740–2748, December 2015. 4

Nonparametric Active learning under local smoothness assumption

Boris Ndjia Njike *[1] et Xavier Siebert †[1]

[1]Department of Mathematics and Operational Research, Faculté Polytechnique, Université de Mons, Rue de Houdain 9, 7000 Mons, Belgium.

June 1, 2019

Abstract

There is a large body of work on convergence rates either in passive or active learning. Here we first outline some of the main results that have been obtained, more specifically in a nonparametric setting under assumptions about the smoothness and the margin noise. We discuss the relative merits of these underlying assumptions by putting active learning in perspective with recent work on passive learning. We provide a novel active learning algorithm with a rate of convergence better than in passive learning, using a particular smoothness assumption customized for k-nearest neighbors. This smoothness assumption provides a dependence on the marginal distribution of the instance space unlike those are commonly used in the recent literature. Our algorithm thus avoids the strong density assumption that supposes the existence of the density function of the marginal distribution of the instance space and is therefore more generally applicable.

Keywords: Nonparametric learning, active learning, nearest-neighbors, smoothness condition.

1 Introduction

Active learning is a machine learning approach for reducing the data labelling effort. Given an instance space $\mathcal{X}$ or a pool of unlabelled data $\{X_1, \ldots, X_w\}$ provided by a distribution P_X, the learner focuses its labeling effort only on the most "informative" points so that a model built from them can achieve the best possible guarantees [6]. Such guarantees are particularly interesting when they are significantly better than those obtained in passive learning [10]. In the context of this work, we consider binary classification (where the label Y of X takes its value in $\{0, 1\}$) in a nonparametric setting. Extensions to multiclass classification and adaptive algorithms are discussed in the last section.

The nonparametric setting has the advantage of providing guarantees with many informations such as the dependence on the dimensional and distributional parameters by using some hypotheses on the regularity of the decision boundary [4], on the regression function [20, 15], and on the geometry of instance space (called strong density assumption) [1, 15, 20]. One of the initial works on nonparametric active learning [4] assumed that the decision boundary is the graph of a smooth function, that a margin assumption very similar to Tsybakov's noise assumption [18] holds, and that distribution P_X is uniform. This led to a better guarantee than in passive learning. Instead of the assumption on the decision boundary, other works [20, 15] supposed rather that the regression function is smooth (in some sense). This assumption, along with Tsybakov's noise assumption and strong density assumption also gave a

*borisedgar.ndjianjike@umons.ac.be
†xavier.siebert@umons.ac.be

better guarantee than in passive learning. Moreover, unlike in [4], they provided algorithms that are adaptive with respect to the margin's noise and to the smoothness parameters.
However, recent work [5] pointed out some disadvantages of the preceding smoothness assumption, and extended it in the context of passive learning with k-nearest neighbors (k-nn) by using another smoothness assumption that is able to sharply characterize the rate of convergence for all probability distributions that satisfy it.

In this paper, we thus extend the work of [5] to the active learning setting, and provide a novel algorithm that outputs a classifier with the same rate of convergence as other recent algorithms that were using more restrictive hypotheses, as for example [20, 15].

Section 2 introduces general definitions and Section 3 presents previous related work on convergence rates in active and passive non-parametric learning, with a special emphasis on the assumptions related to our work. Section 4 describes our algorithm, along with the theoretical motivations and main results. Finally, Section 6 concludes this paper with a discussion of possible extensions of this work.

2 Preliminaries

We begin with some general definitions and notations about active learning in binary classification, then summarize the main assumptions that are typically used to study the rate of convergence of active learning algorithms in the framework of statistical learning theory.

2.1 Active learning setting

Let $(\mathcal{X}, \rho)$ a metric space and $\mathcal{Y} = \{0, 1\}$ the label space. In this paper we set $\mathcal{X} = \mathbb{R}^d$ and refer to it as the instance space, with ρ the Euclidean metric. We assume that the couples (X, Y) are random variables distributed according to an unknown probability P over $\mathcal{X} \times \mathcal{Y}$. Let us denote P_X the marginal distribution of P over $\mathcal{X}$.

Given $w \in \mathbb{N}$ and an i.i.d sample $(X_1, Y_1), \ldots, (X_w, Y_w)$ drawn according to probability P, the learning problem consists in minimizing the risk $\mathcal{R}(f) = P(Y \neq f(X))$ over all measurable functions, called classifiers $f : \mathcal{X} \to \mathcal{Y}$.

In active learning, the labels are not available from the beginning but we can request iteratively at a certain cost (to a so-called oracle) a given number n of samples, called the budget ($n \leq w$). In passive learning, all labels are available from the beginning, and $n = w$. At any time, we choose to request the label of a point X according to the previous observations. The point X is chosen to be most "informative", which amounts to belonging to a region where classification is difficult and requires more labeled data to be collected. Therefore, the goal of active learning is to design a sampling strategy that outputs a classifier $\widehat{f}_n$ whose excess risk is as small as possible with high probability over the requested samples, as reviewed in [6, 10, 7].

Given x in $\mathcal{X}$, let us introduce $\eta(x) = \mathbb{E}(Y|X = x) = P(Y = 1|\, X = x)$ the regression function. As done in [17], it is easy to show that the function $f^*(x) = \mathbb{1}_{\eta(x) \geq 1/2}$ achieves the minimum risk and that $\mathcal{R}(f^*) = \mathbb{E}_X(\min(\eta(X), 1 - \eta(X)))$. Because P is unknown, the function f^* is unreachable and thus the aim of a learning algorithm is to return a classifier $\widehat{f}_n$ with minimum excess risk $\mathcal{R}(\widehat{f}_n) - \mathcal{R}(f^*)$ with high probability over the sample $(X_1, Y_1), \ldots, (X_n, Y_n)$.

2.2 k nearest neighbors (k-nn) classifier

Given two integers k, n such that $k < n$, and a test point $X \in \mathcal{X}$, the k-nn classifier predicts the label of X by giving the majority vote of its k nearest neighbors amongst the sample $X_1, \ldots, X_n$. For $k = 1$, the k-nn classifier returns the label of the nearest neighbor of X amongst the sample $X_1, \ldots, X_n$. Often k grows with n, in which case the method is called k_n-nn. For a complete discussion of nearest neighbors classification, see for example [3, 5].

2.3 Regularity, noise and strong density assumptions

Let $B(x,r) = \{x' \in \mathcal{X},\ \rho(x,x') \leq r\}$ and $B^o(x,r) = \{x' \in \mathcal{X},\ \rho(x,x') < r\}$ the closed and open balls (with respect to the Euclidean metric ρ), respectively, centered at $x \in \mathcal{X}$ with radius $r > 0$. Let $\text{supp}(P_X) = \{x \in \mathcal{X}\ \forall r > 0,\ P_X(B(x,r)) > 0\}$ the support of the marginal distribution P_X.

Definition 1 (Hölder-continuity)**.**
*Let $\eta : \mathcal{X} \to [0,1]$ the regression function. We say that η is α-**Hölder continuous** $(0 < \alpha \leq 1)$ if $\forall$ $x, x' \in \mathcal{X}$,*

$$|\eta(x) - \eta(x')| \leq L\rho(x,x')^\alpha, \tag{H1}$$

for some constant $L > 0$.

The notion of Hölder continuity ensures that the proximity between two closest (according to the metric ρ) points is reflected in a similar value for the conditional probability $\eta(x)$.
This definition remains true for a general metric space, but for the case where ρ is the Euclidean metric, we should always have $0 < \alpha \leq 1$, otherwise η becomes constant.

Definition 2 (Strong density)**.**
*Let P the distribution probability defined over $\mathcal{X} \times \mathcal{Y}$ and P_X the marginal distribution of P over $\mathcal{X}$. We say that P satisfies the **strong density** assumption if there exists some constants $r_0 > 0$, $c_0 > 0$, $p_{min} > 0$ such that for all $x \in supp(P_X)$:*

$$\begin{aligned} &\lambda(B(x,r) \cap supp(P_X)) \geq c_0 \lambda(B(x,r)),\ \forall r \leq r_0 \\ &and\, p_X(x) > p_{min}. \end{aligned} \tag{H2}$$

where p_X is the density function of the marginal distribution P_X and λ is the Lebesgue measure.

The strong density assumption ensures that, given a realisation $X = x$ according to P_X, there exists an infinite number of realisations $X_1 = x_1, \ldots, X_m = x_m, \ldots$ in a neighborhood of x.

Definition 3 (Margin noise)**.**
*We say that P satisfies **margin noise** or **Tsybakov's noise** assumption with parameter $\beta \geq 0$ if for all $0 < \epsilon \leq 1$*

$$P_X(x \in \mathcal{X},\ |\eta(x) - 1/2| < \epsilon) < C\epsilon^\beta, \tag{H3}$$

for $C := C(\beta) \in [1, +\infty[$.

The margin noise assumption gives a bound on the probability that the label of the training points in the neigborhood of a test point x differs from the label of x given by the conditional probability $\eta(x)$. It also describes the behavior of the regression function in the vicinity of the decision boundary $\eta(x) = \frac{1}{2}$. When β goes to infinity, we observe a "jump" of η around to the decision boundary, and then we obtain Massart's noise condition [19]. Small values of β allow for η to "cuddle" $\frac{1}{2}$ when we approach the decision boundary.

Definition 4 (α-smooth)**.**
*Let $0 < \alpha \leq 1$. The regression function is α-**smooth** if for all $x, z \in supp(P_X)$ we have:*

$$\begin{aligned} &|\eta(x) - \eta(z)| \leq L.P_X(B^o(x, \rho(x,z)))^{\alpha/d} \\ &for\ some\ constant\ L \geq 1. \end{aligned} \tag{H4}$$

Theorem 1 states that the α-smooth assumption (H4) is more general than the Hölder continuity assumption (H1).

Theorem 1. *[5]*
Suppose that $\mathcal{X} = \mathbb{R}^d$, that the regression function η is α_h-Hölder continuous, and that P_X has a density with respect to Lebesgue measure greater or equal to $\geq p_{min}$. Then there is a constant $L > 1$ such that for any $x, z \in supp(P_X)$, and $r > 0$, we have:

$$|\eta(x) - \eta(z)| \leq L.P_X(B^o(x, \rho(x,z)))^{\alpha_h/d}.$$

3 Convergence rates in nonparametric active learning

3.1 Previous works

Active learning theory has been mostly studied during the last decades in a parametric setting, see for example [2, 11, 7] and references therein. One of the pioneering works studying the achievable limits in active learning in a nonparametric setting [4] required that the decision boundary is the graph of a Hölder continuous function with parameter α (H1). Using a notion of margin's noise (with parameter β) very similar to (H3), the following minimax rate was obtained:

$$O\left(n^{-\frac{\beta}{2\beta+\gamma-2}}\right), \tag{1}$$

where $\gamma = \frac{d-1}{\alpha}$ and d is the dimension of instance space ($\mathcal{X} = \mathbb{R}^d$).

Note that this result assumes the knowledge of smoothness and margin's noise parameters, whereas an algorithm that achieves the same rate, but that adapts to these parameters was proposed recently in [16].

In this paper, we consider the case where the smoothness assumption refers to the regression function both in passive and in active learning.

In passive learning, by assuming that the regression function is Hölder continuous (H1), along with (H3) and (H2), the minimax rate was established by [1] :

$$O\left(n^{-\frac{\alpha(\beta+1)}{2\alpha+d}}\right). \tag{2}$$

In active learning, using the same assumptions (H1), (H3) and (H2), with the additional condition $\alpha\beta < d$, the following minimax rate was obtained [15] :

$$\tilde{O}\left(n^{-\frac{\alpha(\beta+1)}{2\alpha+d-\alpha\beta}}\right), \tag{3}$$

where $\tilde{O}$ indicates that there may be additional logarithmic factors. This active learning rate given by (3) thus represents an improvement over the passive learning rate (2) that uses the same hypotheses.

With another assumption on the regression function relating the L_2 and L_∞ approximation losses of certain piecewise constant or polynomial approximations of η in the vicinity of the decision boundary, the same rate (3) was also obtained by [20].

3.2 Link with k-nn classifiers

For practicals applications, an interesting question is if k-nn classifiers attain the rate given by (2) in passive learning and by (3) in active learning.

In passive learning, under assumptions (H1), (H3) and (H2), and for suitable k_n, it was shown in [5] that k_n-nn indeed achieves the rate (2).

In active learning a pool-based algorithm that outputs a k-nn classifier has been proposed in [14], but its assumptions differ from ours in terms of smoothness and noise, and the number of queries is constant. Similarly, the algorithm proposed in [9] outputs a 1-nn classifier based on a subsample of the initial pool, such that the label of each instance of this subsample is determined with high probability by the labels of its neighbors. The number of neighbors is adaptively chosen for each instance in the subsample, leading to the minimax rate (3) under the same assumptions as in [15].

To obtain more general results on the rate of convergence for k-nn classifiers in metric spaces under minimal assumptions, the more general smoothness assumption given by (H4) was used in [5]. By using a k-nn algorithm, and under assumptions (H3) and (H4), the rate of convergence obtained in [5] is also on the order of (2).

Additionally, this rate avoids the strong density assumption (H2) and therefore allows more classes of probability. In addition, the α-smooth assumption is more universal than Hölder continuity assumption.

It just holds for any pair of distributions P_X and η. In Hölder continuity, strong density assumption implicitly assumes the existence of the density p_X, and according to (H2), it also implies that the support of P_X has finite Lebesgue measure; this is very restrictive and excludes important densities like Gaussian densities as noticed in [8].

3.3 Contributions of the current work

In this paper, we provide an active learning algorithm under the assumptions (H4) and (H3) that were used in passive learning in [5]. The α-smooth assumption (H4) involves a dependence on the marginal distribution P_X unlike the Hölder continuity assumption (H1).

In the following, we will show that the rate of convergence of our algorithm remains the same as (3), despite the use of more general hypotheses.

4 KALLS algorithm

4.1 Setting

As explained in Section 2.1, we consider a pool of i.i.d unlabeled examples $\mathcal{K} = \{X_1, X_2, \ldots, X_w\}$. Let $n \leq w$ the budget, that is the maximum number of points whose label we are allowed to query to the oracle. The objective of the algorithm is to build a subsample $\{X_{t_i}\}$ whose labels are considered most "informative", and which we call the *active set*. More precisely, a point X_{t_i} is considered "informative" if its label cannot be inferred from the previous observations X_{t_j}(with $t_j < t_i$). The sequence $(t_i)_{i\geq 1}$ of indices is an increasing sequence of integers, starting arbitrarily with $X_{t_1} = X_1$ and stopping when the budget n is attained or when $X_{t_i} = X_w$ for some t_i.

When a point $\{X_{t_i}\}$ is considered informative, instead of requesting its label, we request the labels of its nearest neighbors, as was done in [9]. This differs from the setting of [16], where the label of X_{t_i} is requested several times. This is reasonable for pratical situations where the uncertainty about the label of X_{t_i} has to be overcome, and it is related to the α-smooth assumption (H4). The number of neighbors k_{t_i} is adaptively determined such that (with high confidence) while respecting the budget, we can predict the true label $f^*(X_{t_i})$ of X_{t_i} by

$$\widehat{Y}_{t_i} = \frac{1}{k_{t_i}} \sum_{j \in N_{k_{t_i}}(X_{t_i})} Y_j,$$

where $N_{k_{t_i}}(X_{t_i})$ is the set containing the indices of the k_{t_i}-nearest neighbors of $X(t_i)$.

The final active set output by the algorithm will thus be $\widehat{S}_n = \{(X_{t_1}, \widehat{Y}_{t_1}), \ldots, X_{\widehat{t}_l}, \widehat{Y}_{t_l})\}$ with $\widehat{t}_l \leq w$. This set $\widehat{S}_n$ is obtained as a subset of $\widehat{S}$, which is the set of points considered to be "informative" by removing the points that are too noisy and thus that require many more labels. We show that the active set $\widehat{S}_n$ is sufficient to predict the label of any new point by a 1-nn classification rule $\widehat{f}$.

4.2 Algorithm

Below we provide a description of the KALLS algorithm (Algorithm 1), that aims at determining the active set defined in Section 4.1 and the related 1-nn classifier $\widehat{f}_n$ under the assumptions (H4) and (H3). The complete proofs of the convergence of the algorithm are in Section 5.

For the algorithm KALLS, the inputs are a pool $\mathcal{K}$ of unlabelled data of size w, the budget n, the smoothness parameters α and L from (H4), the margin noise parameters β, C from (H3), a confidence parameter $\delta \in (0, 1)$ and an accuracy parameter $\epsilon \in (0, 1)$. For the moment, these parameters are fixed from the beginning. Adaptive algorithms such as [15] could be exploited, in particular for the α and β parameters. Some constants also appear in the algorithm, as discussed in Section 4.6.

The final active set is obtained such that, with high confidence, the 1-nn classifier $\widehat{f}_n$ based on it agrees with the Bayes classifier at points that lie beyond some margin $\widehat{\Delta}$ of the decision boundary.

Formally, given $x \in \mathcal{X}$ such that $|\eta(x) - 1/2| > \widehat{\Delta}$, we have $\widehat{f}(x) = \mathbb{1}_{\eta(x) \geq 1/2}$ with high confidence. We will show that, with a suitable choice of $\widehat{\Delta}$, the hypothesis (H3) leads to the desired rate of convergence (3).

Algorithm 1: k-nn Active Learning under Local Smoothness (`KALLS`)

Input: $\mathcal{K} = \{X_1, \ldots, X_w\}$, n, α, L, δ, C, β, ϵ, d;
Output: 1-nn classifier $\widehat{f}_n$

$s = 1$ ▷ index of point currently examined
$\widehat{\mathcal{S}} = \emptyset$ ▷ current active set
$u = 0$ ▷ counter for number of requested labels
$I = \emptyset$ ▷ set of indices of informative points
$\widehat{\Delta} = \max(\frac{\epsilon}{2}, \left(\frac{\epsilon}{2C}\right)^{\frac{1}{\beta+1}})$;
while $u \leq n$ *and* $s < w$ **do**
  T=`Reliable`(δ,α,s,L,I,d)
  if T=*True* **then**
    $s = s + 1$
  else
    $I = I \cup \{s\}$
    $[\widehat{Y}, Q_s]$=`ConfidentLabel`(δ, α, s, L, u, $\widehat{\Delta}$)
    $\widehat{\mathcal{S}} = \widehat{\mathcal{S}} \cup \{(X_s, \widehat{Y}, Q_s)\}$; ;
$\widehat{f}_n \leftarrow$ `Learn` $(\widehat{\mathcal{S}})$

`KALLS` (Algorithm 1) uses two main subroutines : `Reliable` and `ConfidentLabel`, which are detailed below (Sections 4.3 and 4.4, respectively). It finally outputs a 1-nn classifier $\widehat{f}_n$ using a subroutine called `Learn`.

4.3 `Reliable` subroutine

The `Reliable` subroutine is a binary test about the point X_s currently considered, to verify if the label of X_s can be inferred with high confidence using the labels of the points currently in the active set. If it is the case, the point X_s is not considered to be informative, its label is not requested and it is not added to the active set.

Algorithm 2: `Reliable` subroutine

Input: δ, α, s, L, I, d
Output: T

for $s' \in I$ **do**
  $\widehat{\zeta}_{s'} = \left| \frac{1}{|Q_{s'}|} \sum_{(X,Y) \in Q_{s'}} Y - \frac{1}{2} \right| - \pi_\delta(|Q_{s'}|, s')$
  $\widehat{p}_{s'} =$ `EstProb` $\left(s, \rho(X_s, X_{s'}), \left(\frac{c_3 \widehat{\zeta}_{s'}}{L}\right)^{d/\alpha}, 50, \frac{\delta}{4c_1 s^2}\right)$
if $\exists\ s' \in I$ *such that* $\widehat{\zeta}_{s'} > \bar{c}\pi_\delta(|Q_{s'}|, s')$ *and* $\widehat{p}_{s'} \leq \frac{75}{94}\left(\frac{c_3}{L}\widehat{\zeta}_{s'}\right)^{d/\alpha}$ **then**
  $T = True$
else
  $T = False$

The `Reliable` subroutine uses `EstProb`$(s, r, \theta, 50, \delta')$ as follows:

1. Call the subroutine `BerEst`$(\theta, \delta', 50)$ and use the random variable p_i as a copy of the Bernoulli variable: $\mathbb{1}_{X \in B(X_s, r)}$.
2. Draw a single p_i, sampled randomly from $\mathcal{K}$.

Algorithm 3: Bernoulli Estimation (BerEst)

Input: accuracy parameter θ, confidence δ', budget parameter b
Output: $\widehat{p}$

Sample $p_1, \ldots, p_4$ ▷ with respect to $\sim p$
$S = \{p_1, \ldots, p_4\}$
$K = \frac{4b}{\theta} \log(\frac{8b}{\delta' \theta})$
for $i = 3 : \log_2(b \log(2K/\delta')/\theta)$ **do**
  $m = 2^i$
  $S = S \cup \{p_{m/2+1}, \ldots, p_m\}$
  $\widehat{p} = \dfrac{1}{m} \displaystyle\sum_{j=1}^{m} p_j$
  if $\widehat{p} > b \log(2m/\delta')/m$ **then**
    Break
Output $\widehat{p}$

4.4 `ConfidentLabel` subroutine

If the point X_s is considered informative, the `ConfidentLabel` subroutine is used to determine with a given level of confidence, the label of the current point X_s. This is done by using the labels of its closest k_s nearest neighbors, where k_s is chosen such that, with high probability, the empirical majority of k_s labels differs from the majority in expectation by less than some margin, and all the k_s nearest neighbors are at most at some distance from X_s.

When a point X_s is relatively far away from the decision boundary, the subroutine provides a lower confidence bound $O(\widehat{\zeta_s}) \leq |\eta(X_s) - \frac{1}{2}|$, where $\widehat{\zeta_s}$ is defined in Section 5.3.4. A new point X_t is considered uninformative if we have a low degree of uncertainty on its label, in other words if $|\eta(X_t) - \frac{1}{2}|$ entails the same confidence lower bound $O(\widehat{\zeta_s})$ for some previous informative point X_s.

Using the smoothness assumption, it suffices to have $P_X(B(X_t, \rho(X_t, X_s)) \leq O((\widehat{\zeta_s})^{d/\alpha})$. Because P_X is unknown, we use the subroutine `BerEst` to adaptively estimate with high probability (over the data) $P_X(B(X_t, \rho(X_t, X_s))$ up to $O((\widehat{\zeta_s})^{d/\alpha})$.

Algorithm 4: `ConfidentLabel` subroutine

Input: δ, α, L,u, s, $\widehat{\Delta}$
Output: $\widehat{Y_s}$, Q_s

$Q_s = \emptyset$
$k = 1$
while $k \leq \min(k_s(\epsilon, \delta), n - u)$ **do**
  if $\left| \frac{1}{k} \sum_{i=1}^{k} Y^{(i)} - \frac{1}{2} \right| > c_4 \pi_\delta(k, s)$ **then**
    exit ▷ cut-off condition
  else
    Request the label Y_s^k of X_s^k
    $Q_s = Q_s \cup \{(X_s^k, Y_s^k)\}$
    $k = k + 1$
    $u = u + 1$
$\widehat{\eta} \leftarrow \frac{1}{|Q_s|} \sum_{(X,Y) \in Q_s} Y$
$\widehat{Y} = \mathbb{1}_{\widehat{\eta} \geq 1/2}$

4.5 `Learn` subroutine

The `Learn` subroutine takes as input the set of points that were considered informative, and returns a subset of those by discarding the noisy points, i.e. where we do not have a sufficient guarantee about the noise $\eta(X)$. After discarding the noisy points, we apply the passive learning on the remaining set by using the 1-nn classifier.

Algorithm 5: `Learn` subroutine

Input: $\widehat{\mathcal{S}}$
Output: $\widehat{f}_n$

$\widehat{\mathcal{S}}_n = \emptyset$
for $\{(X_{s'}, \widehat{Y}, Q)\} \in \widehat{\mathcal{S}}$ **do**
  if $\left| \frac{1}{|Q_s|} \sum_{(X,Y) \in Q_s} Y - \frac{1}{2} \right| \geq \pi_\delta(|Q_s|, s)$ **then**
    $\widehat{\mathcal{S}}_n = \widehat{\mathcal{S}}_n \cup \{(X, \widehat{Y})\}$
$\widehat{f}_n \leftarrow$ the 1-nn classifier on $\widehat{\mathcal{S}}_n$

4.6 Constants in KALLS

The KALLS algorithm uses some constants upon which the theoretical proofs (Section 5) impose some constrains. Let us fix $c_0, c_3, c_6, c_7, \bar{c} \in (0,1)$, $c_5 > 1$. Set $\widehat{c} = \frac{8(1+c_5)^4}{c_4^2 c_0^2}$ and $c_4 =$. We have the following constraints:

$$\begin{cases} 1 \geq c_4 \geq c_0, \\ \frac{c_0}{1+c_5} + \frac{c_0}{c_5} + \frac{c_0}{c_5} \leq 1, \\ c_5^4 \geq \frac{10}{e} c_0^2. \end{cases}$$

For $\epsilon, \delta \in (0,1)$, $k, s \geq 1$, set:

$$\pi_\delta(k,s) = \sqrt{\frac{2}{k}\left(\log\left(\frac{4c_1 s^2}{\delta}\right) + \log\log\left(\frac{4c_1 s^2}{\delta}\right) + \log\log(ek)\right)}. \tag{4}$$

For $\epsilon, \delta \in (0,1)$, $s \geq 1$,

$$k_s(\epsilon,\delta) = \frac{\widehat{c}}{\widehat{\Delta}^2}\left[\log(\frac{4c_1 s^2}{\delta}) + \log\log(\frac{4c_1 s^2}{\delta}) + \log\log\left(\frac{2\sqrt{e}c_5^2}{c_0 c_4 \widehat{\Delta}}\right)\right]. \tag{5}$$

For $X_s \in \mathcal{K} = \{X_1, \ldots, X_w\}$, we denote henceforth by X_s^k its k-th nearest neighbor in $\mathcal{K}$, and Y_s^k the corresponding label.

For an integer $k \geq 1$, let

$$\widehat{\eta}_k(X_s) = \frac{1}{k}\sum_{i=1}^{k} Y_s^{(i)}, \quad \bar{\eta}_k(X_s) = \frac{1}{k}\sum_{i=1}^{k} \eta(X_s^{(i)}). \tag{6}$$

5 Theoretical motivations

This Section provides the main theoretical motivations behind the KALLS algorithm, and is organized as follows:

Section 5.2 outlines the main ideas of the proof, in Section 5.3 we adaptively determine the number of label requests needed to accurately predict the label of an informative point that is relatively far from the boundary decision. In Section 5.3.4, we provide some lemmas that illustrate a sufficient condition for a point to be informative, in Section 5.3.5, we give theorems that allow us to classify each instance relatively far from the decision boundary. Finally in Section **??**, we provide the label complexity and establish Theorem 3.

5.1 Notations

Some notations will be used throughout the proofs are listed here for convenience.

As defined in Section 2.2, let $B(x,r) = \{x' \in \mathcal{X},\ \rho(x,x') \leq r\}$ and $B^o(x,r) = \{x' \in \mathcal{X},\ \rho(x,x') < r\}$ the closed and open balls (with respect to the Euclidean metric ρ), respectively, centered at $x \in \mathcal{X}$ with radius $r > 0$. Let $\text{supp}(P_X) = \{x \in \mathcal{X}\ \forall r > 0,\ P_X(B(x,r)) > 0\}$ the support of the marginal distribution P_X.

Let us denote $\mathcal{A}_a$ the set of active learning algorithms, and $\mathcal{P}(\alpha,\beta)$ the set of probabilities that satisfy the hypotheses (H4) and (H3), where α is the parameter in (H4) and β in (H3).

For $p \in (0,1]$, and $x \in supp(P_X)$, let us define $r_p(x) = \inf\{r > 0, P_X(B(x,r)) \geq p\}$.

5.2 Main ideas of the proof

For a classifier $\widehat{f}_n$, it is well known [17] that the excess of risk is:

$$R(\widehat{f}_n) - R(f^*) = \int_{\{x,\, \widehat{f}_n(x) \neq f^*(x)\}} |2\eta(x) - 1| dP_X(x). \tag{7}$$

Theorem 2 is the main result of this paper, which provides bounds on the excess risk for the `KALLS` algorithm.

Theorem 2 (Excess risk for the `KALLS` algorithm.)**.**
Let the set $\mathcal{P}(\alpha, \beta)$ such that $\alpha\beta < d$ where d is the dimension of the input space $\mathcal{X} \subset \mathbb{R}^d$. Then, we have:

$$\inf_{A \in \mathcal{A}_a} \sup_{P \in \mathcal{P}(\alpha,\beta)} \mathbb{E}_n \left[R(\widehat{f}_n) - R(f^*) \right] \leq \tilde{O}\left(n^{\frac{\alpha(\beta+1)}{2\alpha+d-\alpha\beta}} \right), \tag{8}$$

where $\mathbb{E}_n$ is with respect to the randomness of the algorithm $A \in \mathcal{A}_a$.

The result (8) is also be stated below (Theorem 3) in a more practical form using label complexity (9). This latter form will be used in the proof.

Theorem 3 (Label complexity for the `KALLS` algorithm.)**.**
Let the set $\mathcal{P}(\alpha, \beta)$ such that $\alpha\beta < d$. Let ϵ, $\delta \in (0, 1)$. There exists $n \in \mathbb{N}$ such that:

$$\text{if} \quad n \geq \tilde{O}\left(\left(\frac{1}{\epsilon} \right)^{\frac{2\alpha+d-\alpha\beta}{\alpha(\beta+1)}} \right), \tag{9}$$

then with probability at least $1 - \delta$, we have:

$$\inf_{A \in \mathcal{A}_a} \sup_{P \in \mathcal{P}(\alpha,\beta)} \left[R(\widehat{f}_n) - R(f^*) \right] \leq \epsilon.$$

We thus aim to proof that (9) is a sufficient condition such that with probability $\geq 1 - \delta$, $\widehat{f}_n$ agrees with f^* on $\{x,\ |\eta(x) - 1/2| > \widehat{\Delta}\}$, for some $\widehat{\Delta} > 0$. Introducing $\widehat{\Delta}$ in (8) leads to:

$$R(\widehat{f}_n) - R(f^*) \leq 2\widehat{\Delta} P_X(|\eta(x) - 1/2| < \widehat{\Delta}).$$

Therefore, if $\widehat{\Delta} \leq \frac{\epsilon}{2}$ then, $R(\widehat{f}_n) - R(f^*) \leq \epsilon$. Otherwise, if $\widehat{\Delta} > \frac{\epsilon}{2}$, by the hypothesis (H3), we have $R(\widehat{f}_n) - R(f^*) \leq 2C\widehat{\Delta}^{\beta+1}$. In the latter case, setting $\widehat{\Delta} = \left(\frac{\epsilon}{2C}\right)^{\frac{1}{\beta+1}}$ guarantees $R(\widehat{f}_n) - R(f^*) \leq \epsilon$. Altogether, the value $\widehat{\Delta} = \max(\frac{\epsilon}{2}, \left(\frac{\epsilon}{2C}\right)^{\frac{1}{\beta+1}})$ guarantees $R(\widehat{f}_n) - R(f^*) \leq \epsilon$.

5.3 First Part: adaptive label requests on informative points

Let us recall that the notations used throughout this Section are defined in Section 5.1.

5.3.1 Known results

Lemma 1 (Chernoff, [21])**.**
Suppose $X_1, \ldots, X_m$ are independent random variables taking value in $\{0, 1\}$. Let X denote their sum and $\mu = E(X)$ its expected value. Then, for any $\delta > 0$,

$$P_m(X \leq (1 - \delta)\mu) \leq \exp(-\delta^2 \mu/2),$$

where P_m is the probability with respect to the sample $X_1, \ldots, X_m$.

Lemma 2 (Logarithmic relationship, [22]).
Suppose $a, b, c > 0$, $abe^{c/a} > 4\log_2(e)$, and $u \geq 1$. Then:

$$u \geq 2c + 2a\log(ab) \Rightarrow \; u > c + a\log(bu).$$

Lemma 3 (Chauduri and Dasgupta, [5]).
For all $p \in (0, 1]$, and $x \in supp(P_X)$, we have:

$$P_X(B(x, r_p(x)) \geq p.$$

5.3.2 Core theorem

Theorem 4 (Adaptive label requests on informative points).
Let $\epsilon, \delta \in (0, 1)$. Set $\widehat{\Delta} = \max(\epsilon, \left(\frac{\epsilon}{2C}\right)^{\frac{1}{\beta+1}})$, and $p_\epsilon = \left(\frac{c_0(1-c_0)\widehat{\Delta}}{L}\right)^{d/\alpha}$, where α, β, L, C are parameters used in (H3) *and* (H4)*, and c_0 the constant introduced in Section 4.6*

For $k_s := k_s(\epsilon, \delta)$ defined in (5)*, Section 4.6 and introducing, for $k, s \geq 1$*

$$\tau_{k,s} = \sqrt{\frac{2}{k}\log(\frac{4c_1 s^2}{\delta})}.$$

There exists an event A_1 with probability at least $1 - \frac{\delta}{2c_1}$, such that on A_1, for all $1 \leq s \leq w$, if

$$k_s \leq (1 - \tau_{k_s,s})p_\epsilon(w - 1) \tag{10}$$

then the k_s nearest neighbors of X_s (in the pool $\mathcal{K}$) belong to the ball $B(X_s, r_{p_\epsilon}(X_s))$. Additionnally, the condition

$$w \geq \tilde{O}\left(\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d}{\alpha(\beta+1)}}\right) \tag{11}$$

is sufficient to have (10).

Proof.
Fix $x \in supp(P_X)$. For $k \in \mathbb{N}$, let us denote $X_x^{(k)}$, the kth nearest neighbor of x in the pool. we have,

$$P(\rho(x, X_x^{(k_s+1)}) > r_{p_\epsilon}(x)) \leq P(\sum_{i=1}^{w} \mathbb{1}_{X_i \in B(x, r_{p_\epsilon}(x))} \leq k_s).$$

Then, by using Lemma 1 and Lemma 3, and if k_s satisfies (10), we have:

$$\begin{aligned}
P(\rho(x, X_x^{(k_s+1)}) > r_{p_\epsilon}(x)) &\leq P(\sum_{i=1}^{w} \mathbb{1}_{X_i \in B(x, r_{p_\epsilon}(x))} \leq (1 - \tau_{k_s,s})p_\epsilon(w-1)) \\
&\leq P\left(\sum_{i=1}^{w} \mathbb{1}_{X_i \in B(x, r_{p_\epsilon}(x))} \leq (1 - \tau_{k_s,s})P_X(B(x, r_{p_\epsilon}(x)))(w-1)\right) \\
&\leq \exp(-\tau_{k_s,s}^2(w-1)P_X(B(x, r_{p_\epsilon}(x))/2) \\
&\leq \exp(-\tau_{k_s,s}^2(w-1)p_\epsilon/2) \\
&\leq \exp(-\tau_{k_s,s}^2 k_s/2) \\
&\leq \exp(-\log(4c_1 s^2/\delta)) \\
&= \frac{\delta}{4c_1 s^2}.
\end{aligned}$$

Fix $x = X_s$. Given X_s, there exists an event $A_{1,s}$, such that $P(A_{1,s}) \geq 1 - \delta/(4c_1 s^2)$, and on $A_{1,s}$, if

$$k_s \leq (1 - \tau_{k_s,s})p_\epsilon(w-1),$$

we have $B(X_s, r_{p_\epsilon}(X_s)) \cap \{X_1, \ldots, X_w\} \geq k_s$. By setting $A_1 = \cap_{s \geq 1} A_{1,s}$, we have $P(A_1) \geq 1 - \delta/2c_1$, and on A_1, for all $1 \leq s \leq w$, if $k_s \leq (1 - \tau_{k_s,s})p_\epsilon(w-1)$, then $B(X_s, r_{p_\epsilon}(X_s)) \cap \{X_1, \ldots, X_w\} \geq k_s$.

Now, let us proof that the condition (11) is sufficient to guarantee (10). First, note the relation (10) implies $w \geq \frac{k_s}{(1-\tau_{k_s,s})p_\epsilon} + 1$. Using a bit of calculus, we can see that if $\tau_{k_s,s} \leq \frac{1}{2}$, then

$$\begin{aligned}
\frac{k_s}{(1-\tau_{k_s,s})p_\epsilon} + 1 &\leq \frac{2k_s}{p_\epsilon} + 1 \\
&\leq 4\frac{k_s}{p_\epsilon} \qquad \left(\text{because } \frac{k_s}{p_\epsilon} \geq 1\right) \\
&= \frac{4\widehat{c}}{p_\epsilon \widehat{\Delta}^2}\left[\log(\frac{4c_1 s^2}{\delta}) + \log\log(\frac{4c_1 s^2}{\delta}) + \log\log\left(\frac{2\sqrt{e}c_5^2}{c_0 c_4 \widehat{\Delta}}\right)\right] \\
&= \frac{\widehat{b}}{\widehat{\Delta}^{2+\frac{d}{\alpha}}}\left[\log(\frac{4c_1 s^2}{\delta}) + \log\log(\frac{4c_1 s^2}{\delta}) + \log\log\left(\frac{2\sqrt{e}c_5^2}{c_0 c_4 \widehat{\Delta}}\right)\right] \\
&\qquad \text{where } \widehat{b} = 4\widehat{c}\left(\frac{L}{c_0(1-c_0)}\right)^{d/\alpha} \\
&\leq \bar{C}\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d}{\alpha(\beta+1)}}\left[\log(\frac{4c_1 s^2}{\delta}) + \log\log(\frac{4c_1 s^2}{\delta}) + \log\log\left(\frac{2\sqrt{e}c_5^2}{c_0 c_4 \widehat{\Delta}}\right)\right] \\
&\qquad \text{as } \widehat{\Delta} = \max(\epsilon, \left(\frac{\epsilon}{2C}\right)^{\frac{1}{\beta+1}}), \text{ where } \bar{C} = \widehat{b}(2C)^{\frac{2\alpha+d}{\alpha(\beta+1)}} \\
&\leq \bar{C}\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d}{\alpha(\beta+1)}}\left[2\log(\frac{4c_1 s^2}{\delta}) + \log\left(\frac{2\sqrt{e}c_5^2}{c_0 c_4 \epsilon}\right)\right] \\
&\qquad \text{as} \log(x) \leq x, \text{ and } \widehat{\Delta} \geq \epsilon \\
&\leq 2\bar{C}\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d}{\alpha(\beta+1)}}\left[\log(s^2) + \log\left(\frac{8c_1\sqrt{e}c_5^2}{c_0 c_4 \delta\epsilon}\right)\right] \\
&\leq 4\bar{C}\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d}{\alpha(\beta+1)}}\left[\log(s) + \log\left(\frac{8c_1\sqrt{e}c_5^2}{c_0 c_4 \delta\epsilon}\right)\right] \\
&\leq 4\bar{C}\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d}{\alpha(\beta+1)}}\left[\log(w) + \log\left(\frac{8c_1\sqrt{e}c_5^2}{c_0 c_4 \delta\epsilon}\right)\right]
\end{aligned} \tag{12}$$

In lemma 2, let us set the constants a, b, c respectively to

$$a = 4\bar{C}\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d}{\alpha(\beta+1)}}, \qquad b = 1, \qquad c = 4\bar{C}\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d}{\alpha(\beta+1)}} \log\left(\frac{8c_1\sqrt{e}c_5^2}{c_0 c_4 \delta\epsilon}\right)$$

.

This implies $c \geq a$, $a \geq 4$ and 2 this gives :

$$abe^{c/a} \geq 4e > \log_2(e).$$

Then, the relation

$$w \geq 4\bar{C}\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d}{\alpha(\beta+1)}}\left(\log\left(\frac{8c_1\sqrt{e}c_5^2}{c_0 c_4 \delta\epsilon}\right) + \log\left(4\bar{C}\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d}{\alpha(\beta+1)}}\right)\right)$$

is sufficient to guarantee (12), which concludes the proof. □

Let us note that the guarantee (11) obtained in the Theorem 4 corresponds to that obtained in passive setting ($w = n$).

5.3.3 Choice of k_s

In the following we provide a justification for the choice of $k_s(\epsilon, \delta)$ (5). We also prove that, if the budget allows us, under a margin condition, the following cut-off condition used in Algorithm 4

$$\left| \frac{1}{k} \sum_{i=1}^{k} Y^{(i)} - \frac{1}{2} \right| \leq c_4 \pi_\delta(k, s)$$

will be violated only on a number of label requests lower than k_s. The intuition behind this, is with Lemma 6, to adapt (with respect to the noise) the number of label requested. Explicitely, fewer label requests will be used on a less-noisy point, and more label requests on a noisy point. This provides a significant saving in the number of requests needed to predict with high probability the right label.

The following is based on a result from the bandits literature [13], which provides a new deviation inequality for a martingale with sub-Gaussian increments, state here as Lemma 5 and on Hoeffding's inequality (Lemma 4).

Lemma 4 (Hoeffding's inequality)**.** *[12]*
Let X be a random variable with $E(X) = 0$, $a \leq X \leq b$, then for $v > 0$,

$$E(e^{vX}) \leq e^{v^2(b-a)^2/8}.$$

Lemma 5 (Lemma 7 from [13])**.**
Let $\zeta(u) = \sum_{k \geq 1} k^{-u}$. Let $X_1, X_2, \ldots$ be independent random variables, identically distributed, such that, for all $v > 0$, $E(e^{vX_1}) \leq e^{v^2\sigma^2/2}$. For every positive integer t, let $S_t = X_1 + \ldots + X_t$. Then, for all $\gamma > 1$ and $r \geq \frac{8}{(e-1)^2}$:

$$P\left(\bigcup_{t \in \mathbb{N}^*} \left\{ |S_t| > \sqrt{2\sigma^2 t(r + \gamma \log\log(et))} \right\} \right) \leq \sqrt{e}\zeta(\gamma(1 - \frac{1}{2r}))(\frac{\sqrt{r}}{2\sqrt{2}} + 1)^\gamma \exp(-r).$$

Lemma 6.
Let $\delta \in (0, \min(0.53c_1, 1))$. For $s \in \{1, \ldots, w\}$. let us denote $X_s^{(k)}$ the kth nearest neighbor of X_s, and $Y_s^{(k)}$ the corresponding label. For $\widehat{\eta}_k(X_s)$ and $\bar{\eta}_k(X_s)$ defined in (6) and $\pi_\delta(k, s)$ defined in (4), there exists an event A_2, such that $P(A_2) \geq 1 - \delta/2c_1$, and on A_2, for all $s \in \{1, \ldots, w\}$, we have:

$$|\widehat{\eta}_k(X_s) - \bar{\eta}_k(X_s)| \leq \pi_\delta(k, s).$$

Proof.
This proof follows that of Theorem 8 in [13], with few additional modifications.

Let $s \in \{1, \ldots, w\}$. Set $S_k = \sum_{i=1}^{k} \left(Y_s^{(i)} - \eta(X_s^{(i)}) \right)$. Given $\{X_1, \ldots, X_w\}$, $E(Y_s^{(k)} - \eta(X_s^{(k)})) = 0$, and the random variables $\left\{ Y_s^{(i)} - \eta(X_s^{(i)}),\ i = 1, \ldots, k \right\}$ are independent. Then by Lemma 4, given $\{X_1, \ldots, X_w\}$, as $Y_s^{(1)} - \eta(X_s^{(1)})$ takes values in $[-1, 1]$, we have $E(e^{v(Y_s^{(1)} - \eta(X_s^{(1)}))}) \leq e^{v^2/2}$ for all $v > 0$. Furthermore, set $z = \log(\frac{4c_1 s^2}{\delta})$, and $r = z + 3\log(z)$. We have $r \geq \frac{8}{(e-1)^2}$, and by Lemma 5, with $\gamma = 3/2$, we have:

$$\begin{aligned} P\left(\bigcup_{k \in \mathbb{N}^*} \left\{ |S_k| > \sqrt{2k(r + \gamma \log\log(ek))} \right\} \right) &\leq \sqrt{e}\zeta(3/2(1 - \frac{1}{2r}))(\frac{\sqrt{r}}{2\sqrt{2}} + 1)^{3/2} \exp(-r) \\ &= \frac{\sqrt{e}}{8}\zeta\left(\frac{3}{2} - \frac{3}{4(z + 3\log(z))} \right) \frac{(\sqrt{z + 3\log(z)} + \sqrt{8})^{3/2}}{z^3} \frac{\delta}{4c_1 s^2} \end{aligned}$$

It can be shown numerically that for $z = \log(\frac{4c_1 s^2}{\delta}) \geq \log(\frac{4c_1}{\delta}) \geq 2.03$,

$$\frac{\sqrt{e}}{8}\zeta\left(\frac{3}{2} - \frac{3}{4(z+3\log(z))}\right)\frac{(\sqrt{z+3\log(z)}+\sqrt{8})^{3/2}}{z^3} \leq 1.$$

Then, if $\delta < \min(4c_1\exp(-2.03), 1) \simeq \min(0.53c_1, 1)$ and given $s \in \{1, \ldots, w\}$, there exists an event $A_{2,s}$ such that $P(A_{2,s}) \geq 1 - \delta/4c_1 s^2$, and simultaneously for all $k \geq 1$, we have:

$$|S_k| \leq \sqrt{2k\left(\log\left(\frac{4c_1 s^2}{\delta}\right) + \log\log\left(\frac{4c_1 s^2}{\delta}\right) + \log\log(ek)\right)}.$$

By setting $A_2 = \cap_{s\geq 1} A_{2,s}$, we have $P(A_2) \geq 1 - \delta/2c_1$, and on A_2, we have for all $s \in \{1, \ldots, w\}$, for all $k \geq 1$,

$$|\widehat{\eta}_k(X_s) - \bar{\eta}_k(X_s)| \ \leq \pi_\delta(k, s).$$

□

Lemma 7.
Let $m \geq 1$ and $u \geq 20$. Then we have:

$$m \geq 2u\log(\log(u)) \Longrightarrow m \geq u\log(\log(m)).$$

Proof.
Define $\phi(m) = m - u\log(\log(m))$, and let $m_0 = 2u\log(\log(u))$. We have:

$$\begin{aligned}\phi(m_0) &= 2u\log(\log(u)) - u(\log(\log(2u\log(\log(u))))) \\ &= 2u\log(\log(u)) - u\log(\log(2u) + \log(\log(\log(u)))\end{aligned}$$

It can be shown numerically that $\phi(m_0) \geq 0$ for $u \geq 20$.
Also, we have: $\phi'(m) = \frac{m\log(m)-u}{m\log(m)} \geq 0$ for all $m \geq m_0$ (notice that $m_0 \geq u$ for $u \geq 20$). Then it is easy to see that $\phi(m) \geq \phi(m_0)$ for all $m \geq m_0$, which establishes the Lemma. □

Theorem 5.
Let $\delta \in (0, \min(0.53c_1, 1))$, and $\epsilon \in (0, 1)$. Let the constants c_0, c_1, c_4, c_5 defined in Section 4.6, Let us assume that w satisfies (11)
For X_s, set $\tilde{k}_s(\epsilon, \delta)$ as

$$\tilde{k}_s(\epsilon, \delta) = \frac{8(1+c_5)^4}{c_0^2|\eta(X_s) - \frac{1}{2}|^2}\left[\log(\frac{4c_1 s^2}{\delta}) + \log\log(\frac{4c_1 s^2}{\delta}) + \log\log\left(\frac{\sqrt{2e}c_5^2}{c_0|\eta(X_s) - \frac{1}{2}|}\right)\right].$$

Let $\widehat{\Delta} = \max(\frac{\epsilon}{2}, \left(\frac{\epsilon}{2C}\right)^{\frac{1}{\beta+1}})$ and $\pi_\delta(k, s)$ defined in (4).

Let us suppose the budget n is sufficiently large with respect to k_s, for all $s \leq w$ such that the subroutine `ConfidentLabel` *is independent of n.*
Then, there exists an event A_3, such that $P(A_3) \geq 1 - \delta/2c_1$, and on $A_1 \cap A_2 \cap A_3$, we have:
for all $s \leq w$, if $|\eta(X_s) - \frac{1}{2}| \geq c_4\widehat{\Delta}$, then, $\tilde{k}_s(\epsilon, \delta) \leq k_s$, and the subroutine `ConfidentLabel(s)` *uses at most $\tilde{k}_s(\epsilon, \delta)$ label requests, and we have:*

1. $$|\frac{1}{\bar{k}_s}\sum_{i=1}^{\bar{k}_s} Y_s^i - \frac{1}{2}| \geq c_5\pi_\delta(\bar{k}_s, s), \tag{13}$$

   where $\bar{k}_s$ is the number of requests made in `ConfidentLabel(s)`.

2. We also have

$$f^*(X_s) = \mathbb{1}_{\widehat{\eta}_{\bar{k}_s}(X_s) \geq \frac{1}{2}}, \tag{14}$$

where

$$\widehat{\eta}_{\bar{k}_s}(X_s) = \frac{1}{\bar{k}_s} \sum_{i=1}^{\bar{k}_s} Y_s^i.$$

Proof.
For $\widehat{\eta}_k(X_s)$ and $\bar{\eta}_k(X_s)$ defined in (6), given $\{X_1, \ldots, X_w\}$, and given X_s, by Hoeffding's inequality, there exists an event $A_{3,s}$, with $P(A_{3,s}) \geq 1 - \delta/4c_1 s^2$, and on $A_{3,s}$, we have:

$$|\widehat{\eta}_k(X_s) - \bar{\eta}_k(X_s)| \leq \sqrt{\frac{2\log(\frac{4c_1 s^2}{\delta})}{k}}.$$

This implies that:

$$|\widehat{\eta}_k(X_s) - \frac{1}{2}| \geq |\bar{\eta}_k(X_s) - \frac{1}{2}| - \sqrt{\frac{2\log(\frac{4c_1 s^2}{\delta})}{k}}. \tag{15}$$

On the event A_1, we have, for all $k \leq k_s$, by α-smoothness assumption (H4),

$$|\eta(X_s) - \eta(X_s^{(k)})| \leq c_0(1 - c_0)\widehat{\Delta}. \tag{16}$$

And then, if $|\eta(X_s) - \frac{1}{2}| \geq c_4\widehat{\Delta}$, then $|\eta(X_s) - \frac{1}{2}| \geq c_0\widehat{\Delta}$ (as $c_4 \geq c_0$). The expression (16) becomes

$$|\eta(X_s^{(k)}) - \frac{1}{2}| \geq c_0^2|\eta(X_s) - \frac{1}{2}|.$$

Then (15) becomes:

$$|\widehat{\eta}_k(X_s) - \frac{1}{2}| \geq c_0^2|\eta(X_s) - \frac{1}{2}| - \sqrt{\frac{2\log(\frac{4c_1 s^2}{\delta})}{k}}. \tag{17}$$

A sufficient condition for k to satisfy (13), is

$$c_0^2|\eta(X_s) - \frac{1}{2}| - \sqrt{\frac{2\log(\frac{4c_1 s^2}{\delta})}{k}} \geq c_5\pi_{k,s},$$

and then:

$$c_0^2|\eta(X_s) - \frac{1}{2}| - \sqrt{\frac{2\log(\frac{4c_1 s^2}{\delta})}{k}} \geq c_5\sqrt{\frac{2}{k}\left(\log\left(\frac{4c_1 s^2}{\delta}\right) + \log\log\left(\frac{4c_1 s^2}{\delta}\right) + \log\log(ek)\right)}.$$

This implies:

$$k \geq \frac{1}{c_0^2|\eta(X_s) - \frac{1}{2}|^2}\left(\sqrt{2\log(\frac{4c_1 s^2}{\delta})} + c_5\sqrt{2\left(\log\left(\frac{4c_1 s^2}{\delta}\right) + \log\log\left(\frac{4c_1 s^2}{\delta}\right) + \log\log(ek)\right)}\right)^2. \tag{18}$$

On the other hand, the right-hand side is smaller than:

$$\frac{1}{c_0^2|\eta(X_s) - \frac{1}{2}|^2}\left(\sqrt{2\log(\frac{4c_1 s^2}{\delta})} + c_5\sqrt{2\log\left(\frac{4c_1 s^2}{\delta}\right)} + c_5\sqrt{2\log\log\left(\frac{4c_1 s^2}{\delta}\right)} + c_5\sqrt{2\log\log(ek)}\right)^2.$$

To deduce (18), it suffices that the expression into brackets be lower than:

$$\sqrt{k}c_0|\eta(X_s) - \frac{1}{2}|.$$

Then, it suffices to have simultaneously:

$$\sqrt{2\log(\frac{4c_1 s^2}{\delta})} \leq \sqrt{k}\frac{c_0}{(1+c_5)^2}|\eta(X_s) - \frac{1}{2}|,$$

$$\sqrt{2\log\log(\frac{4c_1 s^2}{\delta})} \leq \sqrt{k}\frac{c_0}{c_5^2}|\eta(X_s) - \frac{1}{2}|,$$

$$\sqrt{2\log\log(ek)} \leq \sqrt{k}\frac{c_0}{c_5^2}|\eta(X_s) - \frac{1}{2}|,$$

$$\frac{c_0}{1+c_5} + \frac{c_0}{c_5} + \frac{c_0}{c_5} \leq 1.$$

Equivalently, we have:

$$k \geq \frac{2(1+c_5)^4}{c_0^2|\eta(X_s) - \frac{1}{2}|^2}\log(\frac{4c_1 s^2}{\delta}), \tag{19}$$

$$k \geq \frac{2c_5^4}{c_0^2|\eta(X_s) - \frac{1}{2}|^2}\log\log(\frac{4c_1 s^2}{\delta}), \tag{20}$$

$$k \geq \frac{2c_5^4}{c_0^2|\eta(X_s) - \frac{1}{2}|^2}\log\log(ek), \tag{21}$$

$$\frac{c_0}{1+c_5} + \frac{c_0}{c_5} + \frac{c_0}{c_5} \leq 1. \tag{22}$$

We can apply Lemma 7 in (21) in which $m = ek$ and $u = \frac{2ec_5^4}{c_0^2|\eta(X_s)-\frac{1}{2}|^2}$. We have $m \geq 1$ and $u \geq 20$ (see Section 4.6) and then, a sufficient condition to have (21) is:

$$k \geq \frac{4c_5^4}{c_0^2|\eta(X_s) - \frac{1}{2}|^2}\log\log\left(\frac{2ec_5^4}{c_0^2|\eta(X_s) - \frac{1}{2}|^2}\right),$$

or

$$k \geq \frac{8c_5^4}{c_0^2|\eta(X_s) - \frac{1}{2}|^2}\log\log\left(\frac{\sqrt{2e}c_5^2}{c_0|\eta(X_s) - \frac{1}{2}|}\right). \tag{23}$$

We can easily see that $\tilde{k}_s(\epsilon,\delta)$ satisfies (19), (20), (23). The condition (22) is given in Section 4.6 then

$$|\sum_{i=1}^{\tilde{k}_s} Y_s^i - \frac{1}{2}| \geq c_5\pi_\delta(\tilde{k}_s, s). \tag{24}$$

As $|\eta(X_s) - \frac{1}{2}| \geq c_4\pi_\delta(\tilde{k}_s, s)$, and with the condition on differents constants, we can easily see that $\tilde{k}_s(\epsilon,\delta) \leq k_s(\epsilon,\delta)$. By taking the minimum value $\bar{k}_s(\epsilon,\delta)$ that satisfies (24), we can see that when the budget allows us, the subroutine `ConfidentLabel` requests $\bar{k}_s(\epsilon,\delta)$ labels, and we have:

$$|\frac{1}{\bar{k}_s}\sum_{i=1}^{\bar{k}_s} Y_s^i - \frac{1}{2}| \geq c_5\pi_\delta(\bar{k}_s, s). \tag{25}$$

By setting $A_3 = \cap_{s\geq 1}A_{3,s}$, we have $P(A_3) \geq 1 - \delta/2c_1$, and we can deduce (13).

Finally, we are going to show that (14) holds on the event $A_1 \cap A_2 \cap A_3$. By Lemma 6, we have on A_2, for all $s \leq w$, $k \leq k_s(\epsilon, \delta)$,

$$|\widehat{\eta}(X_s) - \bar{\eta}_k(X_s)| \leq \pi_\delta(k, s).$$

And then, on $A_1 \cap A_2$, we have for all $s \leq w$, $k \leq k_s(\epsilon, \delta)$:

$$\begin{aligned} |\eta(X_s) - \widehat{\eta}_k(X_s)| &\leq |\eta(X_s) - \bar{\eta}_k(X_s)| + |\bar{\eta}(X_s) - \widehat{\eta}_k(X_s)| \\ &\leq c_0(1-c_0)\widehat{\Delta} + \pi_\delta(k, s) \end{aligned} \tag{26}$$

Assume without loss of generality that $\eta(X_s) \geq \frac{1}{2}$, which leads to:

$$\begin{aligned} \widehat{\eta}_{\bar{k}_s} - \frac{1}{2} &= \widehat{\eta}_{\bar{k}_s} - \eta(X_s) + \eta(X_s) - \frac{1}{2} \\ &\geq -|\widehat{\eta}_{\bar{k}_s} - \eta(X_s)| + \eta(X_s) - \frac{1}{2}. \end{aligned} \tag{27}$$

If $\eta(X_s) - \frac{1}{2} \geq c_4\widehat{\Delta}$, with (26), the expression (27) becomes:

$$\begin{aligned} \widehat{\eta}_{\bar{k}_s} - \frac{1}{2} &\geq -c_0(1-c_0)\widehat{\Delta} - \pi_\delta(\bar{k}_s, s) + c_4\widehat{\Delta} \\ &= (c_4 - c_0(1-c_0))\widehat{\Delta} - \pi_\delta(\bar{k}_s, s) \\ &\geq -\pi_\delta(\bar{k}_s, s) \ \ (\text{as } c_4 \geq c_0) \end{aligned} \tag{28}$$

On the other hand, we have by (13),

$$|\widehat{\eta}_{\bar{k}_s} - \frac{1}{2}| \geq c_5\pi_\delta(k, s),$$

that is to say:

$$\widehat{\eta}_{\bar{k}_s} - \frac{1}{2} \geq c_5\pi_\delta(k, s) \quad \text{or} \quad \widehat{\eta}_{\bar{k}_s} - \frac{1}{2} \leq -c_5\pi_\delta(k, s).$$

By (28), as $c_5 > 1$, we have necessarily $\widehat{\eta}_{\bar{k}_s} - \frac{1}{2} \geq c_5\pi_\delta(k, s)$, and then:

$$\widehat{\eta}_{\bar{k}_s} - \frac{1}{2} \geq \max(-c_5\pi_\delta(k, s), c_5\pi_\delta(k, s)) = c_5\pi_\delta(k, s) \geq 0,$$

Thus we can easily deduce (14). □

5.3.4 Second part: sufficient condition to be an informative point

As in Section 4.4, a sufficient condition for a point X_t to be considered as an uninformative point is:

$$P_X(B(X_t, \rho(X_t, X_s))) \leq O(\widehat{\zeta}_s). \tag{29}$$

for some previous informative point X_s (with $\widehat{\zeta}_s > 0$). As P_X is unkown, we provide a computational scheme sufficient to obtain (29).
First we follow the general procedure used in [14] to adaptively estimate the expectation of a Bernoulli random variable. Second, we apply it to the Bernoulli variable $\mathbb{1}_A$ where $A = \{x,\ x \in B(X_t, r)\}$.

Lemma 8. *[14]*
Let $\delta' \in (0,1)$, $\theta > 0$, $b \geq 7$ and set $g(b) = 1 + \frac{8}{3b} + \sqrt{\frac{2}{b}}$. Let $p_1, p_2, \ldots \in \{0,1\}$ be i.i.d Bernoulli random variables with expectation p. Let $\widehat{p}$ be the output of `BerEst`(θ, b, δ')*. There exists an event A', such that $P(A') \geq 1 - \delta'$, and on A', we have:*

1. *If* $\widehat{p} \leq \frac{\theta}{g(b)}$ *then* $p \leq \theta$.
2. *Let* $\psi := \max(\theta, \frac{p}{g(b)})$. *The number of random draws in the* `BerEst` *subroutine (Algorithm 10) is at most* $\frac{8b\log(\frac{8b}{\delta'\psi})}{\psi}$.

Lemma 9.
Let $\delta \in (0,1)$, $r > 0$. *As previously, for* $k \geq 1$, $s \leq w$, *let be defined*

$$\pi_\delta(k,s) = \sqrt{\frac{2}{k}\left(\log\left(\frac{4c_1 s^2}{\delta}\right) + \log\log\left(\frac{4c_1 s^2}{\delta}\right) + \log\log(ek)\right)}.$$

For $s \in I$, *(where* I *is the set of informative points defined in* `KALLS`*), set*

$$\widehat{\zeta}_s = \left|\frac{1}{|Q_s|}\sum_{(X,Y)\in Q_s} Y - \frac{1}{2}\right| - \pi_\delta(|Q_s|, s)$$

where Q_s *is defined in subroutine* `ConfidentLabel` *(Algorithm 4). Let us assume that* w *satisfies:*

$$w \geq \frac{400\log\left(\frac{1600c_1 w^2}{\delta(c_3\bar{c}\sigma)^{d/\alpha}}\right)}{(c_3\bar{c}\sigma)^{d/\alpha}} \tag{30}$$

where

$$\sigma = \sqrt{\frac{1}{n}\left(\log\left(\frac{4c_1}{\delta}\right) + \log\log\left(\frac{4c_1}{\delta}\right)\right)}, \quad n \text{ the label budget}$$

and c_1, c_2, $\bar{c}$ c_3 *are defined in Section 4.6*
There exists an event A_4, *such that* $P(A_4) \geq 1 - \delta/2c_1$, *we have, on* A_4, *for all* $s \leq w$:
If there exists $1 \leq s' \leq s$ *and* $s' \in I$, *such that:*

$$\widehat{\zeta}_{s'} \geq \bar{c}\pi_\delta(|Q_{s'}|, s') \text{ and } \widehat{p}_s \leq \frac{75}{94}\left(\frac{c_3}{L}\widehat{\zeta}'_s\right)^{d/\alpha}$$

(*where* $\widehat{p}_s :=$ `EstProb`$(s, \rho(X_s, X_{s'}), \left(\frac{c_3}{L}\widehat{\zeta}_{s'}\right)^{d/\alpha}, 50, \delta/4s^2c_1\ )$, *then*

$$P_X(B(X_s, \rho(X_s, X_{s'}))) \leq \left(\frac{c_3}{L}\widehat{\zeta}_{s'}\right)^{d/\alpha}. \tag{31}$$

Proof.
By following the scheme of subroutine `EstProb`, it is a direct application of Lemma 8 by taking for all $s \leq w$, $b = 50$, $\theta = \left(\frac{c_3}{L}\widehat{\zeta}_s\right)^{d/\alpha}$, $\delta' = \delta/4s^2c_1$, $r = \rho(X_s, X_{s'})$, $A_{4,s} := A'$. And then, if we set $A_4 = \cap_{s\geq 1} A_{4,s}$, we have $P(A_4) \geq 1 - \delta/2c_1$, and (31) follows immediatly.
On the other hand, for all $s \leq w$, the number of draws in `EstProb`$(s, \rho(X_s, X_{s'}), \left(\frac{c_3}{L}\widehat{\zeta}_{s'}\right)^{d/\alpha}, 50, \delta/4s^2c_1)$ is always lower than w. Indeed, by Lemma 8, the number of draws is at most:

$$N := \frac{400\log(\frac{400*4s^2c_1}{\delta\psi})}{\psi} \quad \text{where} \quad \psi = \max((\frac{c_3}{L}\widehat{\zeta}_{s'})^{d/\alpha}, \frac{75}{94}P_X(B(X_s, \rho(X_s, X_{s'})))).$$

Then we have:

$$
\begin{aligned}
N &\leq \frac{400 \log\left(\frac{1600 s^2 c_1}{\delta(\frac{c_3}{L}\widehat{\zeta}_{s'})^{d/\alpha}}\right)}{(\frac{c_3}{L}\widehat{\zeta}_{s'})^{\frac{d}{\alpha}}} \\
&\leq \frac{400 \log\left(\frac{1600 s^2 c_1}{\delta(\frac{c_3}{L}\bar{c}\pi_\delta(|Q_{s'}|,s'))^{d/\alpha}}\right)}{(\frac{c_3}{L}\bar{c}\pi_\delta(|Q_{s'}|,s'))^{d/\alpha}} \qquad (\text{as } \widehat{\zeta}_{s'} \geq \bar{c}\pi_\delta(|Q_{s'}|,s')) \\
&\leq \frac{400 \log\left(\frac{1600 c_1 w^2}{\delta(\frac{c_3}{L}\bar{c}\sigma)^{d/\alpha}}\right)}{(\frac{c_3}{L}\bar{c}\sigma)^{d/\alpha}} \qquad (\text{we can easily see that } \pi_\delta(|Q_{s'}|,s') \geq \sigma) \\
&\leq w \qquad (\text{by } (30)).
\end{aligned}
$$

□

5.3.5 Label the instance space

Lemma 10. *Let* $T = \frac{1}{\tilde{p}_\epsilon}\ln(\frac{c_6}{\delta})$, *and* $\tilde{p}_\epsilon = \left(\frac{c_7\widehat{\Delta}}{L}\right)^{d/\alpha}$. *There exists an event* A_5 *such that* $P(A_5) \geq 1-\delta/c_6$, *and on* A_5, *we have:*

$$
\sup_{x\in supp(P_X)} \min_{\bar{X}\in\{X_1,\ldots,X_{T_{\epsilon,\delta}}\}} P_X(B(x,\rho(\bar{X},x))) \leq \tilde{p}_\epsilon. \tag{32}
$$

Proof. As in above, for $x \in \text{supp}(P_X)$, let us introduce

$$
r_{\tilde{p}_\epsilon}(x) = \inf\{r > 0,\ P_X(B(x,r)) \geq \tilde{p}_\epsilon\}.
$$

By Lemma 3, we have $P_X(B(x, r_{\tilde{p}_\epsilon}(x)) \geq \tilde{p}_\epsilon$. Then each $\bar{X} \in$ belongs to $B(x, r_{\tilde{p}_\epsilon}(x))$ with probability at least $\tilde{p}_\epsilon$. If we denote $\widehat{P}$ the probability over the data, we have:

$$
\begin{aligned}
\widehat{P}(\exists\bar{X} \in \{X_1,\ldots,X_{T_{\epsilon,\delta}}\},\ P_X(B(x,\rho(x,\bar{X})) \leq \tilde{p}_\epsilon) &= 1 - \widehat{P}(\forall\bar{X} \in \{X_1,\ldots,X_{T_{\epsilon,\delta}}\},\ P_X(B(x,\rho(x,\bar{X})) > \tilde{p}_\epsilon) \\
&= 1 - \prod_{i=1}^{T_{\epsilon,\delta}} \widehat{P}(P_X(B(x,\rho(x,X_i)) > \tilde{p}_\epsilon) \\
&\geq 1 - \prod_{i=1}^{T_{\epsilon,\delta}} \widehat{P}(\rho(x,X_i) > r_{\tilde{p}_\epsilon}(x)) \\
&= 1 - \prod_{i=1}^{T_{\epsilon,\delta}} (1 - \widehat{P}(\rho(x,X_i) \leq r_{\tilde{p}_\epsilon}(x))) \\
&\geq 1 - (1-\tilde{p}_\epsilon)^{T_{\epsilon,\delta}} \\
&\geq 1 - \exp(-T_{\epsilon,\delta}\tilde{p}_\epsilon) \\
&= 1 - \delta/c_6,
\end{aligned}
$$

which proofs the lemma. □

Lemma 11.
Let $x \in supp(P_X)$ *such that* $|\eta(x) - \frac{1}{2}| > \widehat{\Delta}$. *Let* I *(used in* `KALLS`*) the set of index of informative points. Set* $s_I = \max\{s, s \in I\}$ *(the last informative point). Let us assume that*

$$
s_I \geq T_{\epsilon,\delta} \tag{33}
$$

and w *satisfies* (30) *and* (11).
We have on $A_1 \cap A_2 \cap A_3 \cap A_4 \cap A_5$, *There exists* $s := s(x) \in I$ *such that:*

1. $|\eta(X_s) - \eta(x)| \leq (1 - c_4)|\eta(x) - \frac{1}{2}|$
2. $f^*(x) = f^*(X_s)$

Lemma 12.
Let $x \in supp(P_X)$ such that $|\eta(x) - \frac{1}{2}| > \widehat{\Delta}$. Let I (used in `KALLS`*) the set of indices of informative points. let us assume that $|I| \geq T_{\epsilon,\delta}$. Let $\mathcal{S}_n$ the final active set use in subroutine* `Learn`*. Let $\widehat{f}_{1NN}$ the output of the subroutine* `Learn`*. Let us assume that w satisfies* (30) *and* (11)*. We have on $A_1 \cap A_2 \cap A_3 \cap A_4 \cap A_5$*

$$\widehat{f}_{1NN}(x) = f^*(x).$$

5.3.6 Label complexity

Lemma 13.
Let us assume that w satisfies (30), (11)*, and $w \geq T_{\epsilon,\delta}$. Then, on $A_1 \cap A_2 \cap A_3 \cap A_4 \cap A_5$, the condition* (9) *is sufficient to guarantee* (33)

Proof.
Let I the subset of $\{1, \ldots, w\}$ used in `KALLS1`, and $s_I = \max\{s,\ s \in I\}$. We consider two cases:

1. **First case:** $s_I = w$: we can easily see that (33) is satisfied. And we have trivially that the condition (**??**) is sufficient to guarantee (33).
2. **Second case:** $s_I < w$: then the total number of label requests up to s_I is:

$$\sum_{s \in I} |Q_s| \tag{34}$$

where Q_s is the output in the subroutine `ConfidentLabel`4. Let $s \in I$. For brevety, let us denote `ConfidentLabel`(n, s) := `ConfidentLabel`$(\delta, \alpha, L, u, s, \widehat{\Delta})$. If $s \neq s_I$. We can see that the subroutine `ConfidentLabel`(n, s) implicitly assumes that the process of label request do not takes into account the constraint related to the budget n (very large budget with respect to $k_s(\epsilon, \delta)$). Then we have:

$$n > \sum_{\substack{s \in I \\ s < s_I}} |Q_s| \tag{35}$$

On the other hand, we want to guarantee the condition (33), for this, necessary for all $s \in I$, such that $s \leq T_{\epsilon,\delta}$, and $s \leq s_I$, at the end of the subroutine, the budget n is not yet reached and then we can replace the relation (35) by

$$n > \sum_{\substack{s \in I \\ s \leq s_I \\ s \leq T_{\epsilon,\delta}}} |Q_s| \tag{36}$$

Then, necessary, (33) holds when (36) holds.
Also, for $s \in I$, by theorem**??**, if we assume that $|\eta(X_s) - \frac{1}{2}| \geq c_4\widehat{\Delta}$, we have that $|Q_s| \leq \tilde{k}_s(\epsilon, \delta)$, and the subroutine terminates when the cut-off condition 13 is satisfied. The left hand side of (36) is equal to:

$$\sum_{\substack{s \in I \\ s \leq s_I \\ s \leq T_{\epsilon,\delta} \\ |\eta(X_s) - \frac{1}{2}| \geq c_4\widehat{\Delta}}} |Q_s| + \sum_{\substack{s \in I \\ s \leq s_I \\ s \leq T_{\epsilon,\delta} \\ |\eta(X_s) - \frac{1}{2}| \leq c_4\widehat{\Delta}}} |Q_s| \tag{37}$$

Firstly, let us consider the first term in (37) and denote it by I. Let us denote by B the event:

$$B_s = \{|\eta(X_s) - \frac{1}{2}| \geq c_4\widehat{\Delta}\}.$$

We have

$$\mathbb{1}_{B_s} = \sum_{j=1}^{m_\epsilon} \mathbb{1}_{B_{s,j}} \tag{38}$$

where

$$B_{s,j} = \{2^{j-1} c_4 \widehat{\Delta} \leq |\eta(X_s) - \frac{1}{2}| \leq 2^j c_4 \widehat{\Delta}\} \quad \text{and } m_\epsilon = \max\left(0, \left\lceil \log_2\left(\frac{1}{c_4 \widehat{\Delta}}\right)\right\rceil\right).$$

Then,

$$\begin{aligned} I &\leq \sum_{\substack{s \in I \\ s \leq s_I \\ s \leq T_{\epsilon,\delta} \\ |\eta(X_s) - \frac{1}{2}| \geq c_4 \widehat{\Delta}}} \tilde{k}_s(\epsilon, \delta) \quad \text{by theorem } \textbf{??} \\ &= \sum_{\substack{s \in I \\ s \leq s_I \\ s \leq T_{\epsilon,\delta}}} \sum_{j=1}^{m_\epsilon} \tilde{k}_s(\epsilon, \delta) \mathbb{1}_{B_{s,j}} \end{aligned} \tag{39}$$

On B_j,

$$\begin{aligned} \tilde{k}_s(\epsilon, \delta) &\leq \frac{\widehat{c}}{2^{2j} \widehat{\Delta}^2} \left[\log(\frac{4c_1 s^2}{\delta}) + \log\log(\frac{4c_1 s^2}{\delta}) + \log\log\left(\frac{2\sqrt{e} c_5^2}{c_0 c_4 2^j \widehat{\Delta}}\right)\right] \\ &\leq \frac{\widehat{c}}{2^{2j} \widehat{\Delta}^2} \left[2\log(\frac{4c_1 s^2}{\delta}) + \log\log\left(\frac{2\sqrt{e} c_5^2}{c_0 c_4 \widehat{\Delta}}\right)\right] \end{aligned} \tag{40}$$

Then (39) becomes:

$$\begin{aligned} I &\leq \frac{\widehat{c}}{\widehat{\Delta}^2} \left[2\log(\frac{4c_1 T_{\epsilon,\delta}^2}{\delta}) + \log\log\left(\frac{2\sqrt{e} c_5^2}{c_0 c_4 \widehat{\Delta}}\right)\right] \sum_{j=1}^{m_\epsilon} 2^{-2j} \sum_{\substack{s \in I \\ s \leq s_I \\ s \leq T_{\epsilon,\delta}}} \mathbb{1}_{B_{s,j}} \\ &\leq \frac{\widehat{c}}{\widehat{\Delta}^2} \left[2\log(\frac{4c_1 T_{\epsilon,\delta}^2}{\delta}) + \log\log\left(\frac{2\sqrt{e} c_5^2}{c_0 c_4 \widehat{\Delta}}\right)\right] \sum_{j=1}^{m_\epsilon} 2^{-2j} \sum_{s \leq T_{\epsilon,\delta}} \mathbb{1}_{B_{s,j}} \end{aligned} \tag{41}$$

By Hoeffding's inequality, there exists an even A_6 such that $P(A_6) \geq 1 - \delta/c_6$, and on A_6, we have for all $j \leq m_\epsilon$,

$$\begin{aligned} \frac{\widehat{c}}{\widehat{\Delta}^2} \sum_{s \leq T_{\epsilon,\delta}} \mathbb{1}_{B_{s,j}} &\leq \frac{\widehat{c}}{\widehat{\Delta}^2} \left(T_{\epsilon,\delta} P_X(x,\ |\eta(x) - \frac{1}{2}| \leq 2^j c_4 \widehat{\Delta}) + T_{\epsilon,\delta} \sqrt{\frac{1}{2T_{\epsilon,\delta}} \log\left(\frac{c_6}{\delta}\right)}\right) \\ &\leq \frac{\widehat{c}}{\widehat{\Delta}^2} \left(T_{\epsilon,\delta} P_X(x,\ |\eta(x) - \frac{1}{2}| \leq 2^j c_4 \widehat{\Delta}) + \sqrt{\frac{T_{\epsilon,\delta}}{2} \log\left(\frac{c_6}{\delta}\right)}\right) \\ &\leq \frac{\widehat{c}}{\widehat{\Delta}^2} \left(T_{\epsilon,\delta} 2^{\beta j} c_4^\beta C \widehat{\Delta}^\beta + \frac{1}{\sqrt{\widetilde{p}_\epsilon}} \log\left(\frac{c_6}{\delta}\right)\right) \quad \text{by H3} \\ &= 2^{\beta j} O\left(\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha + d - \alpha\beta}{\alpha(\beta+1)}}\right) + O\left(\left(\frac{1}{\epsilon}\right)^{\frac{4\alpha + d}{2\alpha(\beta+1)}}\right) \end{aligned}$$

Then, (41) becomes:

$$
\begin{aligned}
I &\leq \tilde{O}\left(\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d-\alpha\beta}{\alpha(\beta+1)}}\right)\sum_{j=1}^{m_\epsilon} 2^{(\beta-2)j} + \tilde{O}\left(\left(\frac{1}{\epsilon}\right)^{\frac{4\alpha+d}{2\alpha(\beta+1)}}\right)\sum_{j=1}^{m_\epsilon} 2^{-2j} \\
&\leq \max\left(\tilde{O}\left(\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d-\alpha\beta}{\alpha(\beta+1)}}\right), \tilde{O}\left(\left(\frac{1}{\epsilon}\right)^{\frac{4\alpha+d}{2\alpha(\beta+1)}}\right)\right) \qquad (42)
\end{aligned}
$$

where $\tilde{O}$ includes the logarithmic terms.
Secondly, by using the same argument as with the term I, the second term II in (37) also satisfies the same relation (42). Then the term in (37) is least than:

$$
\max\left(\tilde{O}\left(\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d-\alpha\beta}{\alpha(\beta+1)}}\right), \tilde{O}\left(\left(\frac{1}{\epsilon}\right)^{\frac{4\alpha+d}{2\alpha(\beta+1)}}\right)\right) \qquad (43)
$$

Then if

$$
n \geq \max\left(\tilde{O}\left(\left(\frac{1}{\epsilon}\right)^{\frac{2\alpha+d-\alpha\beta}{\alpha(\beta+1)}}\right), \tilde{O}\left(\left(\frac{1}{\epsilon}\right)^{\frac{4\alpha+d}{2\alpha(\beta+1)}}\right)\right)
$$

we have that n satisfies (36) , and necessary satisfies (33)

□

6 Conclusion

In this paper we have reviewed the main results for convergence rates in a nonparametric setting, with a special emphasis on the relative merits of the assumptions about the smoothness and the margin noise. By putting active learning in perspective with recent work on passive learning that used a particular smoothness assumption customized for k-nn, we provided a novel active learning algorithm with a rate of convergence comparable to stat-of-the art active learning algorithms, but with less restrictive assumptions. Interesting future directions include an extension to multi-class instead of binary classification. For example, [**?**] provides a step in this direction, since it extends the work of [5] to the context of multiclass, but use a stronger hypothesis (see Definition 2.6 in [**?**]) which should be improved. Adaptive algorithms, i.e. where the parameters α, β describing the smoothness and margin noise are unknown should also be explored in our setting. Previous work in this direction was done in [15]. Practical implementations of the `KALLS` algorithm are underway.

References

[1] Audibert, J.Y., Tsybakov, A.B., et al.: Fast learning rates for plug-in classifiers. The Annals of statistics **35**(2), 608–633 (2007)

[2] Balcan, M.F., Hanneke, S., Vaughan, J.W.: The true sample complexity of active learning. Machine learning **80**(2-3), 111–139 (2010)

[3] Biau, G., Devroye, L.: Lectures on the nearest neighbor method. Springer (2015)

[4] Castro, R.M., Nowak, R.D.: Minimax bounds for active learning. IEEE Transactions on Information Theory **54**(5), 2339–2353 (2008)

[5] Chaudhuri, K., Dasgupta, S.: Rates of convergence for nearest neighbor classification. In: Advances in Neural Information Processing Systems, pp. 3437–3445 (2014)

[6] Dasgupta, S.: Two faces of active learning. Theoretical computer science **412**(19), 1767–1781 (2011)

[7] Dasgupta, S.: Active learning theory. Encyclopedia of Machine Learning and Data Mining pp. 14–19 (2017)

[8] Döring, M., Györfi, L., Walk, H.: Rate of convergence of k-nearest-neighbor classification rule. The Journal of Machine Learning Research **18**(1), 8485–8500 (2017)

[9] Hanneke, S.: Nonparametric active learning, part 1: Smooth regression functions (2018). Http://www.stevehanneke.com/

[10] Hanneke, S., Yang, L.: Minimax analysis of active learning. The Journal of Machine Learning Research **16**(1), 3487–3602 (2015)

[11] Hanneke, S., et al.: Rates of convergence in active learning. The Annals of Statistics **39**(1), 333–361 (2011)

[12] Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association **58**(301), 13–30 (1963)

[13] Kaufmann, E., Cappé, O., Garivier, A.: On the complexity of best-arm identification in multi-armed bandit models. The Journal of Machine Learning Research **17**(1), 1–42 (2016)

[14] Kontorovich, A., Sabato, S., Urner, R.: Active nearest-neighbor learning in metric spaces. In: Advances in Neural Information Processing Systems, pp. 856–864 (2016)

[15] Locatelli, A., Carpentier, A., Kpotufe, S.: Adaptivity to noise parameters in nonparametric active learning. Proceedings of Machine Learning Research vol **65**, 1–34 (2017)

[16] Locatelli, A., Carpentier, A., Kpotufe, S.: An adaptive strategy for active learning with smooth decision boundary. In: Algorithmic Learning Theory, pp. 547–571 (2018)

[17] Lugosi, G.: Pattern classification and learning theory. In: Principles of nonparametric learning, pp. 1–56. Springer (2002)

[18] Mammen, E., Tsybakov, A.B., et al.: Smooth discrimination analysis. The Annals of Statistics **27**(6), 1808–1829 (1999)

[19] Massart, P., Nédélec, É., et al.: Risk bounds for statistical learning. The Annals of Statistics **34**(5), 2326–2366 (2006)

[20] Minsker, S.: Plug-in approach to active learning. Journal of Machine Learning Research **13**(Jan), 67–90 (2012)

[21] Mulzer, W.: Five proofs of chernoff's bound with applications. arXiv preprint arXiv:1801.03365 (2018)

[22] Vidyasagar, M.: Learning and generalisation: with applications to neural networks. Springer Science & Business Media (2013)

Graph-based Clustering under Differential Privacy

Rafael Pinot*[1,2], Anne Morvan,* †[1,2], Florian Yger[1], Cédric Gouy-Pailler[2], et Jamal Atif[1]

[1]Université Paris-Dauphine, PSL Research University, CNRS
[2]CEA, LIST, Université Paris-Saclay

May 28, 2019

Abstract

In this paper, we present the first differentially private clustering method for arbitrary-shaped node clusters in a graph. This algorithm takes as input only an approximate Minimum Spanning Tree (MST) $\mathcal{T}$ released under weight differential privacy constraints from the graph. Then, the underlying nonconvex clustering partition is successfully recovered from cutting optimal cuts on $\mathcal{T}$. As opposed to existing methods, our algorithm is theoretically well-motivated. Experiments support our theoretical findings.

1 Introduction

Weighted graph data is known to be a useful representation data type in many fields, such as bioinformatics or analysis of social, computer and information networks. More generally, a graph can always be built based on the data dissimilarity where points of the dataset are the vertices and weighted edges express "distances" between those objects. For both cases, graph clustering is one of the key tools for understanding the underlying structure in the graph [Sch07]. These clusters can be seen as groups of nodes close in terms of some specific similarity.

Nevertheless, it is critical that the data representation used in machine learning applications protects the private characteristics contained in it. Let us consider an application where one wants to identify groups of similar web pages in the sense of traffic volume *i.e.* web pages with similar audience. In that case, the nodes stand for the websites. The link between two vertices represents the fact that some people consult them both. In such a framework, the web browsing history of an individual is used to set the edge weights. We consider that this history can be a very sensitive information for the user, since he/she could have visited sensible content web pages. Treating such datasets as non-private could lead to leaking information such as his/her political, sexual, or religious preferences. As a standard for data privacy preservation, differential privacy [DMNA06] has been designed: an algorithm is differentially private if, given two close databases, it produces statistically indistinguishable outputs. Since then, its definition has been extended to weighted graphs. Though, machine learning applications ensuring data privacy remain rare, in particular for clustering which encounters severe theoretical and practical limitations. Indeed, some clustering methods lack of theoretical support and most of them restrict the data distribution to convex-shaped clusters [NRS07, BLR08, McS09, Dwo11] or unstructured data [HR13, CYC15]. Hence, the aim of this paper is to offer a theoretically motivated private graph clustering. Moreover, to the best of our knowledge, this is the first weight differentially-private clustering algorithm able to detect clusters with an arbitrary shape for weighted graph data. Our method belongs to the family of Minimum Spanning Tree (MST)-based approaches. An MST represents a useful summary of the graph, and appears to be a natural object to describe it at a lower cost. For clustering purposes, it has the appealing property to help retrieving non-convex shapes [Zah71, ABKY88, GZJ06, MCGA17]. Moreover, they appear to be well-suited for incorporating privacy constraints as will be formally proved in this work.

Our contributions are threefold: 1) We provide the first theoretical justifications of MST-based clustering algorithms. 2) We endow DBMSTCLU algorithm [MCGA17], an MST-based clustering algorithm from the literature, with theoretical guarantees. 3) We introduce a differentially-private version of DBMSTCLU and several results on its privacy/utility tradeoff.

This paper have been published in UAI2018, and comes with supplementary materials. One should thus refer to the supplementary materials for missing proofs.

*Rafael Pinot and Anne Morvan contributed equally.

†Partly supported by the *Direction Générale de l'Armement*.

2 Preliminaries

2.1 Notations

Let $\mathcal{G} = (V, E, w)$ be a simple undirected weighted graph with a vertex set V, an edge set E, and a weight function $w := E \to \mathbb{R}$. One will respectively call the edge set and the node set of a graph $\mathcal{G}$ using the applications $E(\mathcal{G})$ and $V(\mathcal{G})$. Given a node set $S \subset V$, one denotes by $\mathcal{G}_{|S}$ the subgraph induced by S. We call $G = (V, E)$ the topology of the graph, and $\mathcal{W}_E$ denotes the set of all possible weight functions that map E to weights in $\mathbb{R}$. For the remaining of this work, cursive letters are used to represent weighted graphs and straight letters refer to topological arguments. Since graphs are simple, the path $\mathcal{P}_{u-v}$ between two vertices u and v is characterized either as the ordered sequence of vertices $\{u, \ldots, v\}$ or corresponding binding edges depending on the context. We also denote $V_{\mathcal{P}_{u-v}}$ the unordered set of such vertices. Besides, edges e_{ij} denote an edge between nodes i and j. Finally, for all positive integers K, $[K] := \{1, \ldots, K\}$.

2.2 Differential privacy in graphs

As opposed to node-differential privacy [KNRS13] and edge-differential privacy [HLMJ09], both based on the graph topology, the privacy framework considered here is weight-differential privacy where the graph topology $G = (V, E)$ is assumed to be public and the private information to protect is the weight function $w := E \to \mathbb{R}$. Under this model introduced by [Sea16], two graphs are said to be neighbors if they have the same topology, and *close* weight functions. This framework allows one to release an almost minimum spanning tree with weight-approximation error of $O(|V| \log |E|)$ for fixed privacy parameters. Differential privacy is ensured in that case by using the Laplace mechanism on every edge weights to release a spanning tree based on a perturbed version of the weight function. The privacy of the spanning tree construction is thus provided by post-processing (cf. Th. 2.5). However, under a similar privacy setting, [Pin18] recently manages to produce the topology of a tree under differential privacy without relying on the post-processing of a more general mechanism such as the "Laplace mechanism". Their algorithm, called PAMST, privately releases the topology of an almost minimum spanning tree thanks to an iterative use of the "Exponential mechanism" instead. For fixed privacy parameters, the weight approximation error is $O\left(\frac{|V|^2}{|E|} \log |V|\right)$, which outperforms the former method from [Sea16] on arbitrary weighted graphs under weak assumptions on the graph sparseness. Thus, we keep here privacy setting from [Pin18].

Definition 2.1 ([Pin18]). *For any edge set E, two weight functions $w, w' \in \mathcal{W}_E$ are neighboring, denoted $w \sim w'$, if $||w - w'||_\infty := \max_{e \in E} |w(e) - w'(e)| \leq \mu$.*

μ represents the sensitivity of the weight function and should be chosen according to the application and the range of this function. The neighborhood between such graphs is clarified in the following definition.

Definition 2.2. *Let $\mathcal{G} = (V, E, w)$ and $\mathcal{G}' = (V', E', w')$, two weighted graphs, $\mathcal{G}$ and $\mathcal{G}'$ are said to be neighbors if $V = V'$, $E = E'$ and $w \sim w'$.*

The so-called weight-differential privacy for graph algorithms is now formally defined.

Definition 2.3 ([Sea16]). *For any graph topology $G = (V, E)$, let $\mathcal{A}$ be a randomized algorithm that takes as input a weight function $w \in \mathcal{W}_E$. $\mathcal{A}$ is called (ϵ, δ)-differentially private on $G = (V, E)$ if for all pairs of neighboring weight functions $w, w' \in \mathcal{W}_E$, and for any set of possible outputs S, one has*

$$\mathbb{P}[\mathcal{A}(w) \in S] \leq e^{\epsilon} \mathbb{P}[\mathcal{A}(w') \in S] + \delta.$$

If $\mathcal{A}$ is (ϵ, δ)-differentially private on every graph topology in a class $\mathcal{C}$, it is said to be (ϵ, δ)-differentially private on $\mathcal{C}$.

One of the first, and most used differentially private mechanisms is the Laplace mechanism. It is based on the process of releasing a numerical query perturbed by a noise drawn from a centered Laplace distribution scaled to the sensitivity of the query. We present here its graph-based reformulation.

Definition 2.4 (reformulation [DMNA06]). *Given some graph topology $G = (V, E)$, for any $f_G : \mathcal{W}_E \to \mathbb{R}^k$, the sensitivity of the function is defined as $\Delta f_G = \max_{w \sim w' \in \mathcal{W}_E} ||f_G(w) - f_G(w')||_1$.*

Definition 2.5 (reformulation [DR13]). *Given some graph topology $G = (V, E)$, any function $f_G : \mathcal{W}_E \to \mathbb{R}^k$, any $\epsilon > 0$, and $w \in \mathcal{W}_E$, the graph-based Laplace mechanism is $\mathcal{M}_L(w, f_G, \epsilon) = f_G(w) + (Y_1, \ldots, Y_k)$ where Y_i are i.i.d. random variables drawn from Lap$(\Delta f_G/\epsilon)$, and Lap(b) denotes the Laplace distribution with scale b $\left(\text{i.e } \textit{probability density } \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)\right)$.*

Theorem 2.1 ([DMNA06]). *The Laplace mechanism is ϵ-differentially private.*

We define hereafter the graph-based Exponential mechanism. In the sequel we refer to it simply as Exponential mechanism. The Exponential mechanism represents a way of privately answering arbitrary range queries. Given some

range of possible responses to the query $\mathcal{R}$, it is defined according to a utility function $u_G := \mathcal{W}_E \times \mathcal{R} \to \mathbb{R}$, which aims at providing some total preorder on the range $\mathcal{R}$ according to the total order in $\mathbb{R}$. The sensitivity of this function is denoted $\Delta u_G := \max_{r \in \mathcal{R}} \max_{w \sim w' \in \mathcal{W}_E} |u_G(w,r) - u_G(w',r)|$.

Definition 2.6. *Given some graph topology $G = (V, E)$, some output range $\mathcal{R} \subset E$, some privacy parameter $\epsilon > 0$, some utility function $u_G := \mathcal{W}_E \times \mathcal{R} \to \mathbb{R}$, and some $w \in \mathcal{W}_E$ the graph-based Exponential mechanism $\mathcal{M}_{Exp}(G, w, u_G, \mathcal{R}, \epsilon)$ selects and outputs an element $r \in \mathcal{R}$ with probability proportional to* $\exp\left(\frac{\epsilon u_G(w,r)}{2\Delta u_G}\right)$.

The Exponential mechanism defines a distribution on a potentially complex and large range $\mathcal{R}$. As the following theorem states, sampling from such a distribution preserves ϵ-differential privacy.

Theorem 2.2 (reformulation [MT07])**.** *For any non-empty range $\mathcal{R}$, given some graph topology $G = (V, E)$, the graph-based Exponential mechanism preserves ϵ-differential privacy,* i.e *if $w \sim w' \in \mathcal{W}_E$,*

$$\mathbb{P}\left[\mathcal{M}_{Exp}(G, w, u_G, \mathcal{R}, \epsilon) = r\right] \leq e^{\epsilon}\mathbb{P}\left[\mathcal{M}_{Exp}(G, w', u_G, \mathcal{R}, \epsilon) = r\right].$$

Further, Th 2.3 highlights the trade-off between privacy and accuracy for the Exponential mechanism when $0 < |\mathcal{R}| < +\infty$. Th 2.4 presents the ability of differential privacy to comply with composition while Th 2.5 introduces its post-processing property.

Theorem 2.3 (reformulation [DR13])**.** *Given some graph topology $G = (V, E)$, some $w \in \mathcal{W}_E$, some output range $\mathcal{R}$, some privacy parameter $\epsilon > 0$, some utility function $u_G := \mathcal{W}_E \times \mathcal{R} \to \mathbb{R}$, and denoting $OPT_{u_G}(w) = \max_{r \in \mathcal{R}} u_G(w, r)$, one has $\forall\, t \in \mathbb{R}$,*

$$u_G\left(G, w, \mathcal{M}_{Exp}(w, u_G, \mathcal{R}, \epsilon)\right) \leq OPT_{u_G}(w) - \frac{2\Delta u_G}{\epsilon}\left(t + \ln|\mathcal{R}|\right)$$

with probability at most $\exp(-t)$.

Theorem 2.4 ([DKM+06])**.** *For any $\epsilon > 0$, $\delta \geq 0$ the adaptive composition of k (ϵ, δ)-differentially private mechanisms is $(k\epsilon, k\delta)$-differentially private.*

Theorem 2.5 (Post-Processing [DR13])**.** *Let $\mathcal{A} : \mathcal{W}_E \to B$ be a randomized algorithm that is (ϵ, δ)-differentially private, and $h : B \to B'$ a deterministic mapping. Then $h \circ \mathcal{A}$ is (ϵ, δ)-differentially private.*

2.3 Differentially private clustering

Differentially private clustering for unstructured datasets has been first discussed in [NRS07]. This work introduced the first method for differentially private clustering based on the k-means algorithm. Since then most of the work in the field focused on adaptation of this method [BLR08, McS09, Dwo11]. The main drawback of [NRS07] is that it is not able to deal with arbitrary-shaped clusters. This issue has been recently investigated in [HR13] and [CYC15]. They proposed two new methods to find arbitrary-shaped clusters in unstructured datasets respectively based on density clustering and wavelet decomposition. Even though both of them allow one to produce non-convex clusters, they only deal with unstructured datasets and thus are not applicable to node clustering in a graph. Our work focuses on node clustering in a graph under weight-differential privacy. Graph clustering has already been investigated in a topology-based privacy framework [MCB15, NIR16], however, these works do not consider weight-differential privacy. Our work is, to the best of our knowledge, the first attempt to define node clustering in a graph under weight-differential privacy.

3 Differentially private tree based clustering

We aim at producing a private clustering method while providing bounds on the accuracy loss. Our method is an adaptation of an existing clustering algorithm DBMSTCLU. However, to provide theoretical guarantees under differential privacy, one needs to rely on the same kind of guarantees in the non-private setting. [MCGA17] did not bring them in their initial work. Hence, our second contribution is to demonstrate the accuracy of this method, first in the non-private context.

In the following we present 1) the theoretical framework motivating MST-based clustering methods, 2) accuracy guarantees of DBMSTCLU in the non-private setting, 3) PTCLUST our private clustering algorithm, 4) its accuracy under differential privacy constraints.

3.1 Theoretical framework for MST-based clustering methods

MST-based clustering methods, however efficient, lack proper motivation. This Section closes this gap by providing a theoretical framework for MST-based clustering. In the sequel, notations from Section 2.1 are kept. The minimum path distance between two nodes in the graph is defined which enables to explicit our notion of Cluster.

Definition 3.1 (Minimum path distance)**.** *Let be $\mathcal{G} = (V, E, w)$ and u, $v \in V$. The minimum path distance between u and v is*

$$d(u, v) = \min_{\mathcal{P}_{u-v}} \sum_{e \in \mathcal{P}_{u-v}} w(e)$$

with $\mathcal{P}_{u-v}$ a path (edge version) from u to v in $\mathcal{G}$.

Definition 3.2 (Cluster)**.** *Let be $\mathcal{G} = (V, E, w)$, $0 < w(e) \leq 1$ $\forall e \in E$ a graph, (V, d) a metric space based on the minimum path distance d defined on $\mathcal{G}$ and $D \subset V$ a node set. $C \subset D$ is a cluster iff. $|C| > 2$ and $\forall C_1, C_2$ s.t. $C = C_1 \cup C_2$ and $C_1 \cap C_2 = \emptyset$, one has:*

$$\operatorname*{argmin}_{z \in D \setminus C_1} \{ \min_{v \in C_1} d(z, v) \} \subset C_2$$

Assuming that a cluster is built of at least 3 points makes sense since singletons or groups of 2 nodes can be legitimately considered as noise. For simplicity of the proofs, the following theorems hold in the case where noise is neglected. However, they are still valid in the setting where noise is considered as singletons (with each singleton representing a generalized notion of cluster).

Theorem 3.1. *Let be $\mathcal{G} = (V, E, w)$ a graph and $\mathcal{T}$ a minimum spanning tree of $\mathcal{G}$. Let also be C a cluster in the sense of Def. 3.2 and two vertices $v_1, v_2 \in C$. Then, $V_{\mathcal{P}_{v_1-v_2}} \subset C$ with $\mathcal{P}_{v_1-v_2}$ a path from v_1 to v_2 in $\mathcal{G}$, and $V_{\mathcal{P}_{v_1-v_2}}$ the set of vertices contained in $\mathcal{P}_{v_1-v_2}$.*

Proof. Let be $v_1, v_2 \in C$. If v_1 and v_2 are neighbors, the result is trivial. Otherwise, as $\mathcal{T}$ is a tree, there exists a unique path within $\mathcal{T}$ between v_1 and v_2 denoted by $\mathcal{P}_{v_1-v_2} = \{v_1, \ldots, v_2\}$. Let now prove by *reductio ad absurdum* that $V_{\mathcal{P}_{v_1-v_2}} \subset C$. Suppose there is $h \in V_{\mathcal{P}_{v_1-v_2}}$ s.t. $h \notin C$. We will see that it leads to a contradiction. We set C_1 to be the largest connected component (regarding the number of vertices) of $\mathcal{T}$ s.t. $v_1 \in C_1$, and every nodes from C_1 are in C. Because of h's definition, $v_2 \notin C_1$. Let be $C_2 = C \setminus C_1$. $C_2 \neq \emptyset$ since $v_2 \in C_2$. Let be $z^* \in \operatorname*{argmin}_{z \in V \setminus C_1} \{ \min_{v \in C_1} d(z, v) \}$ and $e^* = (z^*, v^*)$ an edge that reaches this minimum. Let us show that $z^* \notin C$. If $z^* \in C$, then two possibilities hold:

1. There is an edge $e_{z^*} \in \mathcal{T}$, s.t. $e_{z^*} = (z^*, z')$ with $z' \in C_1$. This is impossible, otherwise by definition of a connected component, $z^* \in C_1$. Contradiction.

2. For all $e_{z^*} = (z^*, z')$ s.t $z' \in C_1$, one has $e_{z^*} \notin \mathcal{T}$. In particular, $e^* \notin \mathcal{T}$. Since h is the neighbor of C_1 in $\mathcal{G}$, there is also $e_h \in \mathcal{T}$, s.t. $e_h = (h, h')$ with $h' \in C_1$. Once again two possibilities hold:

   (a) $w(e_{z^*}) = \min_{z \in V \setminus C_1} \{ \min_{v \in C_1} d(z, v) \} < w(e_h)$. Then, if we replace e_h by e_{z^*} in $\mathcal{T}$, its total weight decreases. So $\mathcal{T}$ is not a minimum spanning tree. Contradiction.

   (b) $w(e_{z^*}) = w(e_h)$, therefore $h \in \operatorname*{argmin}_{z \in V \setminus C_1} \{ \min_{v \in C_1} d(z, v) \}$. Since $h \notin C$, one gets that $\operatorname*{argmin}_{z \in V \setminus C_1} \{ \min_{v \in C_1} d(z, v) \} \not\subset C_2$. Thus, C is not a cluster. Contradiction.

We proved that $z^* \notin C$. In particular, $z^* \notin C_2$. Then, $\operatorname*{argmin}_{z \in V \setminus C_1} \{ \min_{v \in C_1} d(z, v) \} \not\subset C_2$. Thus, C is not a cluster. Contradiction. Finally $h \in C$ and $V_{\mathcal{P}_{v_1-v_2}} \subset C$. □

This theorem states that, given a graph $\mathcal{G}$, an MST $\mathcal{T}$, and any two nodes of C, every node in the path between them is in C. This means that a cluster can be characterized by a subtree of $\mathcal{T}$. It justifies the use of all MST-based methods for data clustering or node clustering in a graph. All the clustering algorithms based on successively cutting edges in an MST to obtain a subtree forest are meaningful in the sense of Th.3.1. In particular, this theorem holds for the use of DBMSTCLU [MCGA17] presented in Section 3.2.1.

3.2 Deterministic MST-based clustering

This Section introduces DBMSTCLU [MCGA17] that will be adapted to be differentially-private, and provides accuracy results on the recovery of the ground-truth clustering partition.

3.2.1 DBMSTCLU algorithm

Let us consider $\mathcal{T}$ an MST of $\mathcal{G}$, as the unique input of the clustering algorithm DBMSTCLU. The clustering partition results then from successive cuts on $\mathcal{T}$ so that a new cut in $\mathcal{T}$ splits a connected component into two new ones. Each final connected component, a subtree of $\mathcal{T}$, represents a cluster. Initially, $\mathcal{T}$ is one cluster containing all nodes. Then, at each iteration, an edge is cut if some criterion, called *Validity Index of a Clustering Partition* (DBCVI) is improved. This edge is greedily chosen to locally maximize the DBCVI at each step. When no improvement on DBCVI can be further made, the algorithm stops. The DBCVI is defined as the weighted average of all *cluster validity indices* which are based on two positive quantities, the *Dispersion* and the *Separation* of a cluster:

Definition 3.3 (Cluster Dispersion)**.** *The Dispersion of a cluster C_i (*DISP*) is defined as the maximum edge weight*

of C_i. If the cluster is a singleton (i.e. contains only one node), the associated Dispersion is set to 0. More formally:

$$\forall i \in [K],\ \mathrm{DISP}(C_i) = \begin{cases} \max\limits_{j,\ e_j \in C_i} w_j & \text{if } |E(C_i)| \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Definition 3.4 (Cluster Separation)**.** *The Separation of a cluster C_i (SEP) is defined as the minimum distance between the nodes of C_i and the ones of all other clusters $C_j, j \neq i, 1 \leq i, j \leq K, K \neq 1$ where K is the total number of clusters. In practice, it corresponds to the minimum weight among all already cut edges from $\mathcal{T}$ comprising a node from C_i. If $K = 1$, the Separation is set to 1. More formally, with $incCuts(C_i)$ denoting cut edges incident to C_i,*

$$\forall i \in [K],\ \mathrm{SEP}(C_i) = \begin{cases} \min\limits_{j,\ e_j \in incCuts(C_i)} w_j & \text{if } K \neq 1 \\ 1 & \text{otherwise.} \end{cases}$$

Definition 3.5 (Validity Index of a Cluster)**.** *The Validity Index of a cluster C_i is defined as:*

$$V_C(C_i) = \frac{\mathrm{SEP}(C_i) - \mathrm{DISP}(C_i)}{\max(\mathrm{SEP}(C_i), \mathrm{DISP}(C_i))} \in [-1; 1]$$

Definition 3.6 (Validity Index of a Clustering Partition)**.** *The Density-Based Validity Index of a Clustering partition $\Pi = \{C_i\}, 1 \leq i \leq K$, $\mathrm{DBCVI}(\Pi)$ is defined as the weighted average of the Validity Indices of all clusters in the partition where N is the number of vertices.*

$$\mathrm{DBCVI}(\Pi) = \sum_{i=1}^{K} \frac{|C_i|}{N} V_C(C_i) \in [-1, 1]$$

DBMSTCLU is summarized in Algorithm 1: $\mathrm{evaluateCut}(.)$ computes the DBCVI when the cut in parameter is applied to $\mathcal{T}$. Initial DBCVI is set -1. Interested reader could refer to [MCGA17] Section 4. for a complete insight on this notions.

Algorithm 1 DBMSTCLU($\mathcal{T}$)

Input: $\mathcal{T}$, the MST
$dbcvi \leftarrow -1.0$
$clusters \leftarrow \emptyset$
$cut\_list \leftarrow \{E(\mathcal{T})\}$
while $dbcvi < 1.0$ **do**
  $cut\_tp \leftarrow \emptyset$
  $dbcvi\_tp \leftarrow dbcvi$
  for each cut in cut_list **do**
    $newDbcvi = \mathrm{evaluateCut}(\mathcal{T}, cut)$
    if $newDbcvi \geq dbcvi\_tp$ **then**
      $cut\_tp \leftarrow cut$
      $dbcvi\_tp \leftarrow newDbcvi$
  if $cut\_tp \neq \emptyset$ **then**
    $clusters = \mathrm{cut}(clusters, cut\_tp)$
    $dbcvi \leftarrow dbcvi\_tp$
    $cut\_list \leftarrow cut\_list \backslash \{cut\_tp\}$
  else
    break
return $clusters$

3.2.2 DBMSTClu exact clustering recovery proof

In this section, we provide theoretical guarantees for the cluster recovery accuracy of DBMSTClu. Let us first begin by introducing some definitions.

Definition 3.7 (Cut)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K clusters, $\mathcal{T}$ an MST of $\mathcal{G}$. Let denote $(C_i^*)_{i \in [K]}$ the set of the clusters. Then, $Cut_{\mathcal{G}}(\mathcal{T}) := \{e_{kl} \in \mathcal{T} \mid k \in C_i^*,\ l \in C_j^*,\ i, j \in [K]^2,\ i \neq j\}$. In the sequel, for simplicity, we denote $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$ the edge between cluster C_i^* and C_j^*.*

$Cut_{\mathcal{G}}(\mathcal{T})$ is basically the set of effective cuts to perform on $\mathcal{T}$ in order to ensure the exact recovery of the clustering partition. More generally, trees on which $Cut_{\mathcal{G}}(.)$ enables to find the right partition are said to be a partitioning topology.

Definition 3.8 (Partitionning topology)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K clusters $C_1^*, \ldots, C_K^*$. A spanning tree $\mathcal{T}$ of $\mathcal{G}$ is said to have a partitioning topology if $\forall i, j \in [K]$, $i \neq j$, $|\{e = (u, v) \in Cut_{\mathcal{G}}(\mathcal{T}) \mid u \in C_i^*, v \in C_j^*\}| = 1$.*

Def. 3.7 and 3.8 introduce a topological condition on the tree as input of the algorithm. Nevertheless, conditions on weights are necessary too. Hence, we define homogeneous separability which expresses the fact that within a cluster the edge weights are spread in a controlled manner.

Definition 3.9 (Homogeneous separability condition)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$, $s \in E$ and $\mathcal{T}$ a tree of $\mathcal{G}$. $\mathcal{T}$ is said to be homogeneously separable by s, if*

$$\alpha_{\mathcal{T}} \max_{e \in E(\mathcal{T})} w(e) < w(s) \text{ with } \alpha_{\mathcal{T}} = \frac{\max\limits_{e \in E(\mathcal{T})} w(e)}{\min\limits_{e \in E(\mathcal{T})} w(e)} \geq 1.$$

One will write for simplicity that $H_{\mathcal{T}}(s)$ is verified.

Definition 3.10 (Weak homogeneity condition of a Cluster)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K clusters $C_1^*, \ldots, C_K^*$. A given cluster C_i^*, $i \in [K]$, C_i^* is weakly homogeneous if: for all $\mathcal{T}$ an MST of $\mathcal{G}$, and $\forall j \in [K]$,*

$j \neq i$, s.t. $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$, $H_{\mathcal{T}_{|C_i^*}}(e^{(ij)})$ is verified. For simplicity, one denotes $\underline{\alpha}_i = \max_{\mathcal{T} \text{ MST of } \mathcal{G}} \alpha_{\mathcal{T}_{|C_i^*}}$

Definition 3.11 (Strong homogeneity condition of a Cluster)**.** *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K clusters $C_1^*, \ldots, C_K^*$. A given cluster C_i^*, $i \in [K]$, C_i^* is strongly homogeneous if: for all $\mathcal{T}$ a spanning tree (ST) of $\mathcal{G}$, and $\forall j \in [K]$, $j \neq i$, s.t. $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$, $H_{\mathcal{T}_{|C_i^*}}(e^{(ij)})$ is verified. For simplicity, one denotes $\bar{\alpha}_i = \max_{\mathcal{T} \text{ ST of } \mathcal{G}} \alpha_{\mathcal{T}_{|C_i^*}}$*

Weak homogeneity is indeed really natural considering the non-private cases using an MST. Strong homogeneity is more demanding, but still a reachable condition. Section 4 presents an experiment where the graph respects strong homogeneity as well as being organised in arbitrary shaped clusters. We show that the weak homogeneity condition is implied by the strong homogeneity condition.

Proposition 3.1. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K clusters $C_1^*, \ldots, C_K^*$. If a given cluster C_i^*, $i \in [K]$ is strongly homogeneous, then, it is weakly homogeneous.*

Proof. If $\mathcal{T}$ a spanning tree of $\mathcal{G}$, and $\forall j \in [K]$, $j \neq i$, s.t. $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$, $H_{\mathcal{T}_{|C_i^*}}(e^{(ij)})$ is verified, then in particular, it is true for any MST. □

Strong homogeneity condition appears to be naturally more constraining on the edge weights than the weak one. The accuracy of DBMSTCLU is proved under the weak homogeneity condition, while the accuracy of its differentially-private version is only given under the strong homogeneity condition.

Theorem 3.2. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K homogeneous clusters $C_1^*, \ldots, C_K^*$ and $\mathcal{T}$ an MST of $\mathcal{G}$. Let now assume that at step $k < K - 1$, DBMSTClu built $k+1$ subtrees $\mathcal{C}_1, \ldots, \mathcal{C}_{k+1}$ by cutting $e_1, e_2, \ldots, e_k \in E$.*

Then, $Cut_k := Cut_{\mathcal{G}}(\mathcal{T}) \setminus \{e_1, e_2, \ldots, e_k\} \neq \emptyset \implies$ $\text{DBCVI}_{k+1} \geq DBCVI_k$, i.e. if there are still edges in Cut_k, the algorithm will continue to perform some cut.

Theorem 3.3. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K homogeneous clusters $C_1^*, \ldots, C_K^*$ and $\mathcal{T}$ an MST of $\mathcal{G}$.*

Assume now that at step $k < K - 1$, DBMSTClu built $k+1$ subtrees $\mathcal{C}_1, \ldots, \mathcal{C}_{k+1}$ by cutting $e_1, e_2, \ldots, e_k \in E$. We still denote $Cut_k := Cut_{\mathcal{G}}(\mathcal{T}) \backslash \{e_1, e_2, \ldots, e_k\}$.

If $Cut_k \neq \emptyset$ then $\underset{e \in \mathcal{T} \backslash \{e_1, e_2, \ldots, e_k\}}{\operatorname{argmax}} DBCVI_{k+1}(e) \subset Cut_k$ i.e. the cut edge at step $k+1$ is in Cut_k.

Theorem 3.4. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K weakly homogeneous clusters $C_1^*, \ldots, C_K^*$ and $\mathcal{T}$ an MST of $\mathcal{G}$. Let now assume that at step $K - 1$, DBMSTClu built K subtrees $\mathcal{C}_1, \ldots, \mathcal{C}_K$ by cutting $e_1, e_2, \ldots, e_{K-1} \in E$. We still denote $Cut_{K-1} := Cut_{\mathcal{G}}(\mathcal{T}) \backslash \{e_1, e_2, \ldots, e_{K-1}\}$.*

Then, for any $e \in \mathcal{T} \backslash \{e_1, e_2, \ldots, e_{K-1}\}$, $DBCVI_K(e) < DBCVI_{K-1}$ i.e. the algorithm stops: no edge gets cut during step K.

Corollary 3.1. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K weakly homogeneous clusters $C_1^*, \ldots, C_K^*$ and $\mathcal{T}$ an MST of $\mathcal{G}$. $DBMSTClu(\mathcal{T})$ stops after $K - 1$ iterations and the K subtrees produced match exactly the clusters i.e. under homogeneity condition, the algorithm finds automatically the underlying clustering partition.*

Proof. Th. 3.2 and 3.4 ensure that under homogeneity condition on all clusters, the algorithm performs the $K - 1$ distinct cuts within $Cut_{\mathcal{G}}(\mathcal{T})$ and stops afterwards. By definition of $Cut_{\mathcal{G}}(\mathcal{T})$, it means the DBMSTClu correctly builds the K clusters. □

3.3 Private MST-based clustering

This section presents our new node clustering algorithm PTCLUST for weight differential privacy. It relies on a mixed adaptation of PAMST algorithm [Pin18] for recovering a differentially-private MST of a graph and DBMSTCLU.

3.3.1 PAMST algorithm

Given a simple-undirected-weighted graph $\mathcal{G} = (V, E, w)$, PAMST outputs an almost minimal weight spanning tree topology under differential privacy constraints. It relies on a Prim-like MST algorithm, and an iterative use of the graph-based Exponential mechanism. PAMST takes as an input a weighted graph, and a utility function. It outputs the topology of a spanning tree which weight is almost minimal. Algorithm 2 presents this new method, using the following utility function:

$$\begin{array}{rcccl} u_G & : & \mathcal{W}_E \times \mathcal{R} & \to & \mathbb{R} \\ & & (w, r) & \mapsto & -|w(r) - \min_{r' \in \mathcal{R}} w(r')|. \end{array}$$

PAMST starts by choosing an arbitrary node to construct iteratively the tree topology. At every iteration, it uses the Exponential mechanism to find the next edge to be added to the current tree topology while keeping the weights private. This algorithm is the state of the art to find a spanning tree topology under differential privacy. For readability, let us introduce some additional notations. Let S be a set of nodes from G, and $\mathcal{R}_S$ the set of edges that are incident to one and only one node in S (also denoted xor-incident). For any edge r in such a set, the incident node to r that is not in S is denoted $r_{\rightarrow}$. Finally, the restriction of the weight function to an edge set $\mathcal{R}$ is denoted $w_{|\mathcal{R}}$.

Algorithm 2 PAMST(G, u_G, w, ϵ)

Input: $\mathcal{G} = (V, E, w)$ a weighted graph (separately the topology G and the weight function w), ϵ a degree of privacy and u_G utility function.
Pick $v \in V$ at random
$S_V \leftarrow \{v\}$
$S_E \leftarrow \emptyset$
while $S_V \neq V$ **do**
  $r = \mathcal{M}_{Exp}(\mathcal{G}, w, u_G, \mathcal{R}_{S_V}, \frac{\epsilon}{|V|-1})$
  $S_V \leftarrow S_V \cup \{r_{\rightarrow}\}$
  $S_E \leftarrow S_E \cup \{r\}$
return S_E

Theorem 3.5 states that using PAMST to get an almost minimal spanning tree topology preserves weight-differential privacy.

Theorem 3.5. *Let $G = (V, E)$ be the topology of a simple-undirected graph, then $\forall \epsilon > 0$,* PAMST$(G, u_G, \bullet, \epsilon)$ *is ϵ-differentially private on G.*

3.3.2 Differentially-private clustering

The overall goal of this Section is to show that one can obtain a differentially-private clustering algorithm by combining PAMST and DBMSTCLU algorithms. However, PAMST does not output a weighted tree which is inappropriate for clustering purposes. To overcome this, one could rely on a sanitizing mechanism such as the Laplace mechanism. Moreover, since DBMSTCLU only takes weights from (0,1], two normalizing parameters τ and p are introduced, respectively to ensure lower and upper bounds to the weights that fit within DBMSTCLU needs. This sanitizing mechanism is called the Weight-Release mechanism. Coupled with PAMST, it will allow us to produce a weighted spanning tree with differential privacy, that will be exploited in our private graph clustering.

Definition 3.12 (Weight-Release mechanism)**.** *Let $\mathcal{G} = (G, w)$ be a weighted graph, $\epsilon > 0$ a privacy parameter, s a scaling parameter, $\tau \geq 0$, and $p \geq 1$ two normalization parameters. The Weight-Release mechanism is defined as*

$$\mathcal{M}_{w.r}(G, w, s, \tau, p) = \left(G, w' = \frac{w + (Y_1, ..., Y_{|E|}) + \tau}{p}\right)$$

where Y_i are i.i.d. random variables drawn from Lap $(0, s)$. With $w + (Y_1, ..., Y_{|E|})$ meaning that if one gives an arbitrary order to the edges $E = (e_i)_{i \in [|E|]}$, one has $\forall i \in [|E|]$, $w'(e_i) = w(e_i) + Y_i$.

The following theorem presents the privacy guarantees of the Weight-Release mechanism.

Theorem 3.6. *Let $G = (V, E)$ be the topology of a simple-undirected graph, $\tau \geq 0$, $p \geq 1$, then $\forall \epsilon > 0$, $\mathcal{M}_{w.r}\left(G, \bullet, \frac{\mu}{\epsilon}, \tau, p\right)$ is ϵ- differentially private on G.*

Proof. Given $\tau \geq 0$, $p \geq 1$, and $\epsilon > 0$, the Weight release mechanism scaled to $\frac{\mu}{\epsilon}$ can be break down into a Laplace mechanism and a post-processing consisting in adding τ to every edge and dividing them by p. Using Theorems 2.1 and 2.5, one gets the expected result. □

So far we have presented DBMSTCLU and PAMST algorithms, and the Weight-Release mechanism. Let us now introduce how to compose those blocks to obtain a Private node clustering in a graph, called PTCLUST. The

Algorithm 3 PTCLUST$(G, w, u_G, \epsilon, \tau, p)$

Input: $\mathcal{G} = (V, E, w)$ a weighted graph (separately the topology G and the weight function w), ϵ a degree of privacy and u_G utility function.
$T =$ PAMST$(G, w, u_G, \epsilon/2)$
$\mathcal{T}' = \mathcal{M}_{w.r}(T, w_{|E(T)}, \frac{2\mu}{\epsilon}, \tau, p)$
return DBMSTCLU$(\mathcal{T}')$

algorithm 3 takes as an input a weighted graph (dissociated topology and weight function), a utility function, a privacy degree and two normalization parameters. It outputs a clustering partition. To do so, a spanning tree topology is produced using PAMST with time and space complexities respectively equal to $O(|V|^2)$ and $O(|E|)$. Afterwards a randomized and normalized version of the associated weight function is released using the Weight-release mechanism. Finally the obtained weighted tree is given as an input to DBMSTCLU that performs a clustering partition with $O(|V|)$ time and space complexities. The following theorem ensures that our method preserves ϵ-differential privacy.

Theorem 3.7. *Let $G = (V, E)$ be the topology of a simple-undirected graph, $\tau \geq 0$, and $p \geq 1$, then $\forall \epsilon > 0$,* PTCLUST$(G, \bullet, u_G, \epsilon, \tau, p)$ *is ϵ-differentially private on G.*

Proof. Using Theorem 3.5 one has that T is produced with $\epsilon/2$-differential privacy, and using Theorem 3.6 one has that w' is obtained with $\epsilon/2$-differential privacy as well. Therefore using Theorem 2.4, $\mathcal{T}'$ is released with ϵ-differential privacy. Using the post-processing property (Theorem 2.5) one gets the expected result. □

3.4 Differential privacy trade-off clustering

The results stated in this section present the security/accuracy trade-off of our new method in the differentially-private framework. PTCLUST relies on two

differentially -private mechanisms, namely PAMST and the Weight-Release mechanism. Evaluating the accuracy of this method amounts to check whether using these methods for ensuring privacy does not deteriorate the final clustering partition. The accuracy is preserved if PAMST outputs the same topology as the MST-based clustering and if the Weight-Release mechanism preserves enough the weight function. According to Def. 3.8, if a tree has a partitioning topology, then it fits the tree-based clustering. The following theorem states that with high probability PAMST outputs a tree with a partitioning topology.

Theorem 3.8. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K strongly homogeneous clusters $C_1^*, \ldots, C_K^*$ and $T = \text{PAMST}(\mathcal{G}, u_{\mathcal{G}}, w, \epsilon)$, $\epsilon > 0$. T has a partitioning topology with probability at least*

$$1 - \sum_{i=1}^{K} (|C_i^*| - 1) \exp\left(-\frac{A}{2\Delta u_{\mathcal{G}}(|V| - 1)}\right)$$

with $A = \epsilon \left(\bar{\alpha}_i \max_{e \in E\left(\mathcal{G}_{|C_i^*}\right)}(w(e)) - \min_{e \in E\left(\mathcal{G}_{|C_i^*}\right)}(w(e)) \right) + \ln |E|.$

The following theorem states that given a tree $\mathcal{T}$ under the strong homogeneity condition, if the subtree associated to a cluster respects Def. 3.9, then it still holds after applying the Weight-Release mechanism to this tree.

Theorem 3.9. *Let us consider a graph $\mathcal{G} = (V, E, w)$ with K strongly homogeneous clusters $C_1^*, \ldots, C_K^*$ and $T = PAMST(\mathcal{G}, u_{\mathcal{G}}, w, \epsilon)$, $\mathcal{T} = (T, w_{|T})$ and $\mathcal{T}' = \mathcal{M}_{w.r}(T, w_{|T}, s, \tau, p)$ with $s \ll p, \tau$. Given some cluster C_i^*, and $j \neq i$ s.t $e^{(ij)} \in Cut_{\mathcal{G}}(\mathcal{T})$, if $H_{\mathcal{T}_{|C_i^*}}(e^{(ij)})$ is verified, then $H_{\mathcal{T}'_{|C_i^*}}(e^{(ij)})$ is verified with probability at least*

$$1 - \frac{\mathbb{V}(\varphi)}{\mathbb{V}(\varphi) + \mathbb{E}(\varphi)^2}$$

with the following notations :

- $\varphi = (\max_{j \in [|C_i^*| - 1]} Y_j)^2 - \min_{j \in [|C_i^*| - 1]} Z_j \times X^{out}$
- $Y_j \underset{iid}{\sim} Lap\left(\frac{\max_{e \in E(\mathcal{T})} w(e) + \tau}{p}, \frac{s}{p}\right)$
- $Z_j \underset{iid}{\sim} Lap\left(\frac{\min_{e \in E(\mathcal{T})} w(e) + \tau}{p}, \frac{s}{p}\right)$
- $X^{out} \sim Lap\left(\frac{w(e^{(ij)}) + \tau}{p}, \frac{s}{p}\right)$,

4 Experiments

So far we have exhibited the trade-off between clustering accuracy and privacy and we experimentally illustrate it with some qualitative results. We have performed experiments on two classical synthetic graph datasets for clustering with nonconvex shapes: two concentric circles and two moons, both in their noisy versions. For the sake of readability and for visualization purposes, both graph datasets are embedded into a two dimensional Euclidean space. Each dataset contains 100 data nodes that are represented by a point of two coordinates. Both graphs have been built with respect to the strong homogeneity condition: edge weights within clusters are between $w_{min} = 0.1$ and $w_{max} = 0.3$ while edges between clusters have a weight strictly above $w_{max}^2 / w_{min} = 0.9$. In practice, the complete graph has been trimmed from its irrelevant edges (*i.e.* not respecting the strong homogeneity condition). Hence, those graphs are not necessarily Euclidean since close nodes in the visual representation may not be connected in the graph. Finally, weights are normalized between 0 and 1.

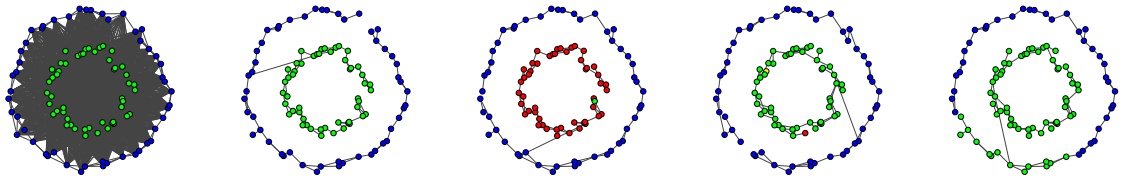
Figures 1 and 2 (best viewed in color) show for each dataset (a) the original homogeneous graph $\mathcal{G}$ built by respecting the homogeneity condition, (b) the clustering partition[1] of DBMSTCLU with the used underlying MST, the clustering partitions for PTCLUST with $\mu = 0.1$ obtained respectively with different privacy degrees[2] : (c) $\epsilon = 0.5$, (d) $\epsilon = 0.7$ and (e) $\epsilon = 1.0$. The utility function $u_{\mathcal{G}}$ corresponds to the graph weight. Each experiment is carried out independently and the tree topology obtained by PAMST will eventually be different. This explains why the edge between clusters may not be the same when the experiment is repeated with a different level of privacy. However, this will marginally affect the overall quality of the clustering.

As expected, DBMSTCLU recovers automatically the right partition and the results are shown here for comparison with PTCLUST. For PTCLUST, the true MST is replaced with a private approximate MST obtained for suitable τ and p ensuring final weights between 0 and 1.

When the privacy degree is moderate ($\epsilon \in \{1.0, 0.7\}$), it appears that the clustering result is slightly affected. More precisely, in Figures 1c and 1d the two main clusters are recovered while one point is isolated as a singleton. This is due to the randomization involved in determining the edge weights for the topology returned by PAMST. In Figure 2c, the clustering is identical to the one from DBMSTCLU in Figure 2b. In Figure 1d, the clustering is very similar to the DBMSTClu one, with the exception of an isolated sin-

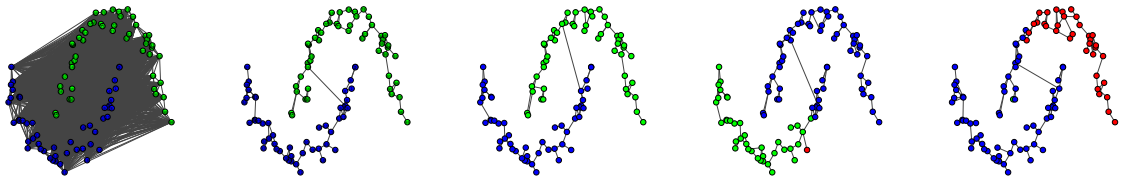
[1] For the sake of clarity, the edges in those Figures are represented based on the original weights and not on the privately released weights.

[2] Note that, although the range of ϵ is in $\mathbb{R}_+^\star$, it is usually chosen in practice in $(0, 1]$ [DR13, Chap 1&2].



(a) Homogeneous graph (b) DBMSTCLU (c) PTCLUST, $\epsilon = 1.0$ (d) PTCLUST, $\epsilon = 0.7$ (e) PTCLUST, $\epsilon = 0.5$

Figure 1: Circles experiments for $n = 100$. PTCLUST parameters: $w_{min} = 0.1$, $w_{max} = 0.3$, $\mu = 0.1$.



(a) Homogeneous graph (b) DBMSTCLU (c) PTCLUST, $\epsilon = 1.0$ (d) PTCLUST, $\epsilon = 0.7$ (e) PTCLUST, $\epsilon = 0.5$

Figure 2: Moons experiments for $n = 100$. PTCLUST parameters: $w_{min} = 0.1$, $w_{max} = 0.3$, $\mu = 0.1$.

gleton. However, as expected from our theoretical results, when ϵ is decreasing (even below 0.5), the clustering quality deteriorates, as DBMSTCLU is sensitive to severe changes in the MST (cf. Figure 1e, 2e).

5 Conclusion

In this paper, we introduced PTCLUST, a novel graph clustering algorithm able to recover arbitrarily-shaped clusters while preserving differential privacy on the weights of the graph. It is based on the release of a private approximate minimum spanning tree of the graph of the dataset, by performing suitable cuts to reveal the clusters. To the best of our knowledge, this is the first differentially private graph-based clustering algorithm adapted to nonconvex clusters. The theoretical analysis exhibited a trade-off between the degree of privacy and the accuracy of the clustering result. Differential privacy is investigated in the framework of strong homogeneity but this is quite restrictive. A smoother result would be very interesting but it is more challenging. This will be a focus of our future work. Our work suits to applications where privacy is a critical issue and it could pave the way to metagenomics and genes classification using individual gene maps while protecting patient privacy. Future work will also be devoted to deeply investigate these applications.

References

[ABKY88] T. Asano, B. Bhattacharya, M. Keil, and F. Yao. Clustering algorithms based on minimum and maximum spanning trees. In *Proceedings of the Fourth Annual Symposium on Computational Geometry*, SCG '88, pages 252–257, New York, NY, USA, 1988. ACM.

[BLR08] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 609–618, New York, NY, USA, 2008. ACM.

[CYC15] L. Chen, T. Yu, and R. Chirkova. Wavecluster with differential privacy. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1011–1020, New York, NY, USA, 2015. ACM.

[DKM+06] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Eurocrypt*, volume 4004, pages 486–503. Springer, 2006.

[DMNA06] C. Dwork, F. McSherry, K. Nissim, and A.Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, pages 265–284. Springer Berlin Heidelberg, 2006.

[DR13] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407, 2013.

[Dwo11] C. Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, January 2011.

[GZJ06] O. Grygorash, Y. Zhou, and Z. Jorgensen. Minimum spanning tree based clustering algorithms. In *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, pages 73–81, Nov 2006.

[HLMJ09] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178, Dec 2009.

[HR13] S.-S. Ho and S. Ruan. Preserving privacy for interesting location pattern mining from trajectory data. *Trans. Data Privacy*, 6(1):87–106, April 2013.

[KNRS13] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. Analyzing graphs with node differential privacy. In *Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography*, TCC'13, pages 457–476, Berlin, Heidelberg, 2013. Springer-Verlag.

[MCB15] Y. Mülle, C. Clifton, and K. Böhm. Privacy-integrated graph clustering through differential privacy. In *EDBT/ICDT Workshops*, 2015.

[MCGA17] A. Morvan, K. Choromanski, C. Gouy-Pailler, and J. Atif. Graph sketching-based massive data clustering. *SIAM Data Mining 2018 (to appear)*, 2017.

[McS09] F. McSherry. Privacy integrated queries. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD)*. Association for Computing Machinery, Inc., June 2009.

[MT07] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Providence, RI, October 2007. IEEE.

[NIR16] H. H. Nguyen, A. Imine, and M. Rusinowitch. Detecting communities under differential privacy. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, WPES '16, pages 83–93, New York, NY, USA, 2016. ACM.

[NRS07] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing - STOC*. ACM Press, 2007.

[Pin18] R. Pinot. Minimum spanning tree release under differential privacy constraints. *ArXiv e-prints*, January 2018.

[Sch07] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27 – 64, 2007.

[Sea16] A. Sealfon. Shortest paths and distances with differential privacy. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems - PODS*. ACM Press, 2016.

[Zah71] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20(1):68–86, January 1971.

On Medians of (Randomized) Pairwise Means

Pierre Laforgue[1], Stephan Clémençon[1], et Patrice Bertail[2]

[1]LTCI, Télécom Paris, Institut Polytechnique de Paris
[2]Modal'X, UPL, Université Paris-Nanterre

19 juin 2019

Résumé

Tournament procedures, recently introduced in [LM16], offer an appealing alternative, from a theoretical perspective at least, to the principle of *Empirical Risk Minimization* in machine learning. Statistical learning by Median-of-Means (MoM) basically consists in segmenting the training data into blocks of equal size and comparing the statistical performance of every pair of candidate decision rules on each data block : that with highest performance on the majority of the blocks is declared as the winner. In the context of nonparametric regression, functions having won all their duels have been shown to outperform empirical risk minimizers w.r.t. the mean squared error under minimal assumptions, while exhibiting robustness properties. It is the purpose of this paper to extend this approach, in order to address other learning problems in particular, for which the performance criterion takes the form of an expectation over pairs of observations rather than over one single observation, as may be the case in pairwise ranking, clustering or metric learning. Precisely, it is proved here that the bounds achieved by MoM are essentially conserved when the blocks are built by means of independent sampling without replacement schemes instead of a simple segmentation. These results are next extended to situations where the risk is related to a pairwise loss function and its empirical counterpart is of the form of a U-statistic. Beyond theoretical results guaranteeing the performance of the learning/estimation methods proposed, some numerical experiments provide empirical evidence of their relevance in practice.

Mots-clef : Median of Means, Tournament procedure, U-statistics, Randomization, Robust estimation.

1 Introduction

In [LM16], the concept of *tournament procedure* for statistical learning has been introduced and analyzed in the context of nonparametric regression, one of the flagship problems of machine learning. The task is to predict a real valued random variable (r.v.) Y based on the observation of a random vector X with marginal distribution $\mu(dx)$, taking its values in $\mathbb{R}^d$ with $d \geq 1$, say by means of a regression function $f : \mathbb{R}^d \to \mathbb{R}$ with minimum expected quadratic risk $\mathcal{R}(f) = \mathbb{E}[(Y - f(X))^2]$. Statistical learning usually relies on a training dataset $\mathcal{S}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ formed of independent copies of the generic pair (X, Y). Following the *Empirical Risk Minimization* (ERM) paradigm, one is encouraged to build predictive rules by minimizing an empirical version of the risk $\hat{\mathcal{R}}_n(f) = (1/n)\sum_{i=1}^n (y_i - f(x_i))^2$ over a class $\mathcal{F}$ of regression function candidates of controlled complexity (*e.g.* Rademacher), while being rich enough to contain a reasonable approximant of the optimal regression function $f^*(x) = \mathbb{E}[Y \mid X = x]$: for any $f \in \mathcal{F}$, the risk excess $\mathcal{R}(f) - \mathcal{R}(f^*)$ is then equal to $||f - f^*||^2_{L_2(\mu)} = \mathbb{E}[(f(X) - f^*(X))^2]$. A completely different learning strategy, recently proposed in [LM16], consists in implementing a *tournament procedure* based on the Median-of-Means (MoM) method, see [NY83]. Precisely, the full dataset is first divided into 3 subsamples of (roughly) equal size. For every pair of candidate functions $(f_1, f_2) \in \mathcal{F}^2$, the first step consists in computing the MoM estimator of the quantity $\|f_1 - f_2\|_{L_1(\mu)} := \mathbb{E}[|f_1(X) - f_2(X)|]$ based on the first subsample : the latter being segmented into $K \geq 1$ subsets of equal size (approximately), $\|f_1 - f_2\|_{L_1(\mu)}$ is estimated by the median of the collection of estimators formed by its empirical versions computed from each of the K subdatasets. When the MoM estimate is large enough, the match between f_1 and f_2 is allowed. The rationale be-

hind this approach is as follows : if one of the candidate, say f_2, is equal to f^*, and the quantity $\|f_1 - f^*\|_{L_1(\mu)}$ (which is less than $\|f_1 - f^*\|_{L_2(\mu)} = \sqrt{\mathcal{R}(f_1) - \mathcal{R}(f^*)}$) is large, so is its (robust) MoM estimate (much less sensitive to atypical values than sampling averages) with high probability. Therefore, f^* is compared to distant candidates only, against which it should hopefully win its matches. The second step consists in computing the MoM estimator of $\mathcal{R}(f_1) - \mathcal{R}(f_2)$ based on the second subsample for every *distant enough* candidates f_1 and f_2. If a candidate wins (*i.e.* has a smaller empirical risk for more than half of the blocks) all its matches, it is kept for the third round. As said before, f^* should be part of this final pool, denoted by H. Finally, matches involving all pairs of candidates in H are computed, using a third MoM estimate on the third part of the data. A champion *winning* all its matches is either f^* or has a small enough excess risk anyway.

It is the purpose of the present article to extend the MoM-based statistical learning methodology. Firstly, we investigate the impact of randomization in the MoM technique : by randomization, it is meant that data subsets are built through sampling schemes, say simple random sampling without replacement (SRSWoR in abbreviated form) for simplicity, rather than partitioning. Though introducing more variability in the procedure, we provide theoretical and empirical evidence that attractive properties of the original MoM method are essentially preserved by this more flexible variant (in particular, the number of blocks involved in this alternative procedure is arbitrary). Secondly, we consider the application of the tournament approach to other statistical learning problems, namely those involving pairwise loss functions, like popular formulations of ranking, clustering or metric-learning. In this setup, natural statistical versions of the risk of low variance take the form of U-statistics (of degree two), *i.e.* averages over all pairs of observations, see *e.g.* [CLV08]. In this situation, we propose to estimate the risk by the median of U-statistics computed from blocks obtained through data partitioning or sampling. Results showing the accuracy of this strategy, referred to as *Median of (Randomized) Pairwise Means* here, are established and application of this estimation technique to pairwise learning is next investigated from a theoretical perspective and generalization bounds are obtained. The relevance of this approach is also supported by convincing illustrative numerical experiments.

The rest of the paper is organized as follows. Section 2 briefly recalls the main ideas underlying the MoM procedure, its applications to robust machine learning as well as basic concepts pertaining to the theory of U-statistics/processes. In Section 3, the variants of the MoM approach we propose are described at length and theoretical results establishing their statistical performance are stated. Illustrative numerical experiments are displayed in Section 4, while proofs are deferred to the Appendix.

2 Background - Preliminaries

As a first go, we briefly describe the main ideas underlying the tournament procedure for robust machine learning, and next recall basic notions of the theory of U-statistics, as well as crucial results related to their efficient approximation. Here and throughout, the indicator function of any event $\mathcal{E}$ is denoted by $\mathbb{I}\{\mathcal{E}\}$, the variance of any square integrable r.v. Z by $\mathrm{Var}(Z)$, the cardinality of any finite set A by $\#A$. If $(a_1, \ldots, a_n) \in \mathbb{R}^n$, the median of $a_1, \ldots, a_n$ is defined as $a_{\sigma((n+1)/2)}$ when n is odd and $a_{\sigma(n/2)}$ otherwise, σ denoting a permutation of $\{1, \ldots, n\}$ such that $a_{\sigma(1)} \leq \ldots \leq a_{\sigma(n)}$. It is denoted by $\mathrm{median}(a_1, \ldots, a_n)$, and sometimes abbreviated as $\mathrm{med}(a_1, \ldots, a_n)$. The floor and ceiling functions are respectively denoted by $u \in \mathbb{R} \mapsto \lfloor u \rfloor$ and $u \in \mathbb{R} \mapsto \lceil u \rceil$.

2.1 Medians of Means based Statistical Learning

First introduced independently by [NY83], [JVV86], and [AMS99], the Median-of-Means (MoM) estimator is a mean estimator dedicated to real random variables. It is now receiving a great deal of attention in the statistical learning literature, following in the footsteps of the results established in [AC11, Cat12], where mean estimators are studied through the angle of their deviation probabilities, rather than based on their traditional mean square errors, for robustness purposes. Indeed, [DLL+16] showed that the MoM estimator provides an optimal δ-dependent subgaussian mean estimator, under the sole assumption that a second order moment exists. The MoM estimator has later been extended to random vectors, through different generalizations of the median. See for instance [M+15, HS16, LM17]. In [BCBL13], it is used to design robust bandits strategies, while [LO11] and [BJL+15] advocate minimizing a MoM, respectively Catoni, estimate of the expected risk, rather than performing empirical risk minimization, to tackle different learning tasks. More recently, [LM16] introduced a tournament strategy based on the MoM approach. In order to describe it precisely, more notations are needed.

The MoM estimator. Let $\mathcal{S}_n = \{Z_1, \ldots, Z_n\}$ be a sample composed of $n \geq 1$ independent realizations of a square integrable real valued r.v. Z, with expectation θ and finite variance $\mathrm{Var}(Z) = \sigma^2$. Dividing $\mathcal{S}_n$ into K disjoint blocks, each with same cardinality $B = \lfloor n/K \rfloor$ approximately, $\tilde{\theta}_k$ denotes the empirical mean based on the data lying in block k for $k \in \{1, \ldots, K\}$. The MoM estimator $\hat{\theta}_{\mathrm{MoM}}$ of θ is then given by :

$$\hat{\theta}_{\mathrm{MoM}} = \mathrm{median}(\tilde{\theta}_1, \ldots, \tilde{\theta}_K).$$

It offers an appealing alternative to the sample mean $\hat{\theta}_n = (1/n)\sum_{i=1}^n Z_i$, much more robust, *i.e.* less sensitive to the presence of atypical values in the sample. Indeed, while exponential concentration inequalities for the sample mean typically necessitates the Z_i's to be subgaussian, they can be established for the MoM estimator in heavy tail situations, under the sole assumption that the Z_i's are square integrable : for any $\delta \in [e^{1-n/2}, 1)$, choosing $K = \lceil \log(1/\delta) \rceil$ and $B = \lfloor n/K \rfloor$, we have [DLL+16, LM16] :

$$\mathbb{P}\left\{ \left| \hat{\theta}_{\mathrm{MoM}} - \theta \right| > 2\sqrt{2}e\sigma \sqrt{\frac{1 + \log(1/\delta)}{n}} \right\} \leq \delta, \quad (1)$$

The tournament procedure. Placing ourselves in the distribution-free regression framework recalled in the Introduction, recall that it has been shown in [LM16] that under appropriate complexity conditions and in a possibly non sub-Gaussian setup, the tournament procedure outputs a candidate $\hat{f}$ with optimal accuracy/confidence tradeoff, outperforming thus ERM in heavy-tail situations. Namely, there exist c, c_0, and $r > 0$ such that, with probability at least $1 - \exp(c_0 n \min\{1, \sigma^{-2} r^2\})$, it holds both :

$$\|\hat{f} - f^*\|_{L_2} \leq cr, \quad \text{and} \quad \mathcal{R}(\hat{f}) - \mathcal{R}(f^*) \leq (cr)^2,$$

see Theorem 2.10 in [LM16].

2.2 Pairwise Means and U-Statistics

Rather than the mean of an integrable r.v., suppose now that the quantity of interest is of the form $\theta(h) = \mathbb{E}[h(X_1, X_2)]$, where X_1 and X_2 are i.i.d. random vectors, taking their values in some measurable space $\mathcal{X}$ with distribution $F(dx)$ and $h : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a measurable mapping, square integrable w.r.t. $F \otimes F$. For simplicity, we assume that $h(x_1, x_2)$ is symmetric (*i.e.* $h(x_1, x_2) = h(x_2, x_1)$ for all $(x_1, x_2) \in \mathcal{X}^2$). A natural estimator of the parameter $\theta(h)$ based on an i.i.d. sample $\mathcal{S}_n = \{X_1, \ldots, X_n\}$ drawn from F is the average over all pairs

$$U_n(h) = \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} h(X_i, X_j). \quad (2)$$

The quantity (2) is known as the U-statistic of degree two [1], with kernel h, based on the sample $\mathcal{S}_n$. One may refer to [Lee90] for an account of the theory of U-statistics. As may be shown by a Lehmann-Scheffé argument, it is the unbiased estimator of $\theta(h)$ with minimum variance. Setting $h_1(X_1) = \mathbb{E}[h(X_1, X_2) \mid X_1] - \theta(h)$, $h_2(X_1, X_2) = h(X_1, X_2) - \theta(h) - h_1(X_1) - h_1(X_2)$, $\sigma_1^2(h) = \mathrm{Var}(h_1(X_1))$ and $\sigma_2^2(h) = \mathrm{Var}(h_2(X_1, X_2))$ and using the orthogonal decomposition (usually referred to as *second Hoeffding decomposition*, see [Hoe48])

$$U_n(h) - \theta(h) = \frac{2}{n} \sum_{i=1}^n h_1(X_i) + \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} h_2(X_i, X_j),$$

one may easily see that

$$\mathrm{Var}(U_n(h)) = \frac{4\sigma_1^2(h)}{n} + \frac{2\sigma_2^2(h)}{n(n-1)}. \quad (3)$$

Of course, an estimator of the parameter $\theta(h)$ taking the form of an i.i.d. average can be obtained by splitting the dataset into two halves and computing

$$M_n(h) = \frac{1}{\lfloor n/2 \rfloor} \sum_{i=1}^{\lfloor n/2 \rfloor} h(X_i, X_{i+\lfloor n/2 \rfloor}).$$

One can check that its variance, $\mathrm{Var}(M_n(h)) = \sigma^2(h)/\lfloor n/2 \rfloor$, with $\sigma^2(h) = \mathrm{Var}(h(X_1, X_2) = 2\sigma_1^2(h) + \sigma_2^2(h)$, is however significantly larger than (3). Regarding the difficulty of the analysis of the fluctuations of (2) (uniformly over a class of kernels possibly), the reduced variance property has a price : the variables summed up being far from independent, linearization tricks (*i.e.* Hajek/Hoeffding projection) are required to establish statistical guarantees for the minimization of U-statistics. Refer to [CLV08] for further details.

Examples. In machine learning, various empirical performance criteria are of the form of a U-statistic.

1. Let $d \geq n$ and $H : \mathcal{X}^d \to \mathbb{R}$ be measurable, square integrable with respect to $F^{\otimes k}$. The statistic $((n-d)!/n!) \sum_{(i_1, \ldots, i_d)} H(X_{i_1}, \ldots, X_{i_d})$, where the sum is taken over all d-tuples with pairwise distinct coordinates of $(1, \ldots, n)$, is a U-statistic of degree d.

• In clustering, the goal is to find a partition $\mathcal{P}$ of the feature space $\mathcal{X}$ so that pairs of observations independently drawn from a certain distribution F on $\mathcal{X}$ within a same cell of $\mathcal{P}$ are more similar w.r.t. a certain metric $D : \mathcal{X}^2 \to \mathbb{R}_+$ than pairs lying in different cells. Based on an i.i.d. training sample $X_1, \ldots, X_n$, this leads to minimize the U-statistic, referred to as *empirical clustering risk* :

$$\widehat{\mathcal{W}}_n(\mathcal{P}) = \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} D(X_i, X_j) \cdot \Phi_{\mathcal{P}}(X_i, X_j), \tag{4}$$

where $\Phi_{\mathcal{P}}(x, x') = \sum_{\mathcal{C} \in \mathcal{P}} \mathbb{I}\{(x, x') \in \mathcal{C}^2\}$, over a class of partition candidates (see [Clé14]).

• In pairwise ranking, the objective is to learn from independent labeled data $(X_1, Y_1), \ldots, (X_n, Y_n)$ drawn as a generic random pair (X, Y), where the real valued random label Y is assigned to an object described by a r.v. X taking its values in a measurable space $\mathcal{X}$, a ranking rule $r : \mathcal{X}^2 \to \{-1, 0, +1\}$ that permits to predict, among two objects (X, Y) and (X', Y') chosen at random, which one is preferred : (X, Y) is preferred to (X', Y') when $Y > Y'$ and, in this case, one would ideally have $r(X, X') = +1$, the rule r being supposed anti-symmetric (*i.e.* $r(x, x') = -r(x', x)$ for all $(x, x') \in \mathcal{X}^2$) This can be formulated as the problem of minimizing the U-statistic known as the *empirical ranking risk* for a given loss $\ell : \mathbb{R} \to \mathbb{R}^+$ [CLV05] :

$$\widehat{\mathcal{L}}_n(r) = \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} \ell\left(-r(X_i, X_j) \cdot (Y_i - Y_j)\right), \tag{5}$$

Other examples of U-statistics are naturally involved in the formulation of metric/similarity-learning tasks, see [BHS13] or [VCB18]. We also point out that the notion of U-statistic is much more general than that considered above : U-statistics of degree higher than two (*i.e.* associated to kernels with more than two arguments) and based on more than one sample can be defined, see *e.g.* Chapter 14 in [VdV00] for further details. The methods proposed and the results proved in this paper can be straightforwardly extended to this more general framework.

3 Theoretical Results

Mainly motivated by pairwise learning problems such as those mentioned in Subsection 2.2, it is the goal of this section to introduce and study several extensions of the MoM approach for robust statistical learning recalled in Subsection 2.1.

3.1 Medians of Randomized Means

As a first go, we place ourselves in the setup of Subsection 2.1, and use the notations introduced therein. But instead of dividing the dataset into disjoint blocks, an arbitrary number K of blocks, of arbitrary size $B \leq n$, are now formed by sampling without replacement independently from $\mathcal{S}_n$. Each randomized data block $\mathcal{B}_k$, $k \in \{1, \ldots, K\}$, is fully characterized by a random vector $\boldsymbol{\epsilon}_k = (\epsilon_{k,1}, \ldots, \epsilon_{k,n})$, such that $\epsilon_{k,i}$ is equal to 1 if the i-th observation has been selected in the k-th block, and to 0 otherwise. The $\boldsymbol{\epsilon}_k$'s are i.i.d. random vectors, uniformly distributed on the set $\Lambda_{n,B} = \{\boldsymbol{\epsilon} \in \{0,1\}^n : \sum_{i=1}^n \epsilon_i = B\}$ of cardinality $\binom{n}{B}$. Equipped with this notation, the empirical mean computed from the k-th randomized block, for $k \in \{1, \ldots, K\}$, can be written as :

$$\bar{\theta}_k = \frac{1}{B} \sum_{i=1}^n \epsilon_{k,i} Z_i. \tag{6}$$

The *Median-of-Randomized Means* (MoRM) estimator $\bar{\theta}_{\text{MoRM}}$ is then given by

$$\bar{\theta}_{\text{MoRM}} = \text{median}(\bar{\theta}_1, \ldots, \bar{\theta}_K). \tag{7}$$

We point out that the number K and size $B \leq n$ of the randomized blocks are arbitrary in the MoRM procedure, in contrast with the usual MoM approach, where $B = \lfloor n/K \rfloor$. However, choices for B and K very similar to those leading to (1) lead to an analogous exponential bound, as revealed by Proposition 1's proof. Because the randomized blocks are not independent, the argument used to establish (1) cannot be applied in a straightforward manner to investigate the accuracy of (7). Nevertheless, as can be seen by examining the proof of the result stated below, a concentration inequality can still be derived, using the conditional independence of the draws given the original data, and a closed analytical form for the conditional probabilities $\mathbb{P}\{|\bar{\theta}_k - \theta| > \varepsilon \mid \mathcal{S}_n\}$, *i.e.* expressed as U-statistics of degree B. Refer to the Appendix for the technical proof.

Proposition 1. *Suppose that $Z_1, \ldots, Z_n$ are independent copies of a square integrable r.v. Z with mean θ and variance σ^2. Then, for any $\tau \in (0, 1/2)$, for any $\delta \in [2e^{-8\tau^2 n/9}, 1)$, choosing $K = \lceil \log(2/\delta)/(2(1/2 - \tau)^2) \rceil$ and $B = \lfloor 8\tau^2 n/(9 \log(2/\delta)) \rfloor$, we have :*

$$\mathbb{P}\left\{ \left|\bar{\theta}_{MoRM} - \theta\right| > \frac{3\sqrt{3}\,\sigma}{2\,\tau^{3/2}} \sqrt{\frac{\log(2/\delta)}{n}} \right\} \leq \delta. \tag{8}$$

The bound stated above presents three main differences with (1). Recall first that the number K of randomized blocks is completely arbitrary in the MoRM procedure and may even exceed n. Consequently, it is always possible to build $\lceil \log(2/\delta)/(2\,(1/2-\tau)^2)\rceil$ blocks, and there is no restriction on the range of admissible confidence levels δ due to K. Second, the size B of the blocks can also be chosen completely arbitrarily in $\{1, \ldots, n\}$, and independently from K. Proposition 1 exhibits their respective dependence with respect to δ and n. Still, B needs to be greater than 1, which results in a restriction on the admissible δ's, such as specified. Observe finally that B never exceeds n. Indeed for all $\tau \in (0, 1/2)$, $8\tau^2/(9\log(2/\delta))$ does not exceeds 1 as long as δ is lower than $2\exp(-2/9) \approx 1.6$, which is always true. Third, the proposed bound involves an additional parameter τ, that can be arbitrarily chosen in $(0, 1/2)$. As may be revealed by examination of the proof, the choice of this extra parameter reflects an accuracy/computational cost trade-off : the larger τ, the larger K, the larger the confidence range, the larger B and the lower the constant in (8) as well. Since one can pick K arbitrarily large, τ can be chosen as large as possible in $(0, 1/2)$. This way, one asymptotically achieves a $3\sqrt{6}$ constant factor, which is the same than that obtained in [HS16] for a comparable confidence range. However, the price of such an improvement is the construction of a higher number of blocks in practice (for a comparable number of blocks, the constant in (8) becomes $27\sqrt{2}$).

Remark 1. (Alternative sampling schemes) *We point out that other procedures than the simple random sampling without replacement (SRSWoR) scheme above (e.g. Poisson/Bernoulli, Monte-Carlo sampling) can naturally be considered to build subsets of observations and compute several estimates of the parameter θ. However, the theoretical analysis of such variants is much more challenging, due to possible replications of the same original observations in a data block.*

Beyond theoretical guarantees, the numerical experiments provide strong empirical evidence of the accuracy of the MoM extension proposed above.

3.2 Medians of (Randomized) U-statistics

We now consider the situation described in Subsection 2.2, where the parameter of interest is $\theta(h) = \mathbb{E}[h(X_1, X_2)]$ and investigate the performance of two possible approaches for extending the MoM methodology.

Medians of U-statistics. The most straightforward way of extending the MoM approach is undoubtedly to form complete U-statistics based on K subsamples corresponding to sets of indexes $I_1, \ldots, I_K$ of size $B = \lfloor n/K \rfloor$ built by segmenting the full sample, as originally proposed : for $k \in \{1, \ldots, K\}$,

$$\hat{U}_k(h) = \frac{2}{B(B-1)} \sum_{(i,j)\in I_k^2,\, i<j} h(X_i, X_j). \qquad (9)$$

The *median of U-statistics* estimator (MoU in abbreviated form) of the parameter $\theta(h)$ is then defined as

$$\hat{\theta}_{\mathrm{MoU}}(h) = \mathrm{median}(\hat{U}_1(h), \ldots, \hat{U}_K(h)). \qquad (10)$$

The following result provides a bound analogue to (1) revealing its accuracy.

Proposition 2. *Let $\delta \in [e^{1-2n/9}, 1)$. Choosing $K = 9/2\log(1/\delta)$, we have with probability at least $1-\delta$:*

$$\left|\hat{\theta}_{MoU}(h) - \theta(h)\right| \leq \sqrt{\frac{C_1 \log(\frac{1}{\delta})}{n} + \frac{C_2 \log^2(\frac{1}{\delta})}{n\left(2n - 9\log(\frac{1}{\delta})\right)}},$$

with $C_1 = 108\sigma_1^2(h)$ and $C_2 = 486\sigma_2^2(h)$.

We point out that another robust estimator $\hat{\theta}_{\mathrm{MoM}}(h)$ of θ could also have been obtained by applying the classic Mo(R)M methodology recalled in Subsection 2.1 to the set of $\lfloor n/2 \rfloor$ i.i.d. observations $\{h(X_i, X_{i+\lfloor n/2 \rfloor}) : 1 \leq i \leq \lfloor n/2 \rfloor\}$, see the discussion in Subsection 2.2. In this context, we deduce from Eq. (1) with $K = \lceil \log(1/\delta) \rceil$ and $B = \lfloor \lfloor n/2 \rfloor / K \rfloor$ that, for any $\delta \in [e^{1-\lfloor n/2 \rfloor/2}, 1)$

$$\left|\hat{\theta}_{\mathrm{MoM}}(h) - \theta(h)\right| \leq 2\sqrt{2}e\sigma(h)\sqrt{\frac{1+\log(1/\delta)}{\lfloor n/2 \rfloor}} \qquad (11)$$

with probability at least $1-\delta$. The Mo(R)M strategies on independent pairs lead to constants respectively equal to $4e\sigma(h)$ and $6\sqrt{3}\sigma(h)$. On the other hand, MoU reaches a $6\sqrt{3}\sigma_1(h)$ constant factor on its dominant term. Recalling that $\sigma^2(h) = 2\sigma_1^2(h) + \sigma_2^2(h)$, beyond the $\sqrt{2}$ constant factor, MoU provides an improvement all the more significant that $\sigma_2^2(h)$ is large. Another difference between the bounds is the restriction on δ, which is looser in the MoU case. This is due to the fact that the MoU estimator can possibly involve any pair of observations among the $n(n-1)/2$ possible ones, in contrast to $\hat{\theta}_{\mathrm{MoM}}(h)$ that relies on the $\lfloor n/2 \rfloor$ pairs set once and for all at the beginning only. MoU however exhibits a more complex *two rates* formula,

but the second term being negligible the performance are not affected, as shall be confirmed empirically.

As suggested in Subsection 3.1, the data blocks used to compute the collection of K U-statistics could be formed by means of a SRSWoR scheme. Confidence bounds for such a median of randomized U-statistics estimator, comparable to those achieved by the MoU estimator, are stated below.

Medians of Randomized U-statistics. The alternative we propose consists in building an arbitrary number K of data blocks $\mathcal{B}_1, \ldots, \mathcal{B}_K$ of size $B \leq n$ by means of a SRSWoR scheme, and, for each data block $\mathcal{B}_k$, forming all possible pairs of observations in order to compute

$$\bar{U}_k(h) = \frac{2}{B(B-1)} \sum_{i<j} \boldsymbol{\epsilon}_{k,i}\boldsymbol{\epsilon}_{k,j} \cdot h(X_i, X_j), \qquad (12)$$

where $\boldsymbol{\epsilon}_k$ denotes the random vector characterizing the $k-th$ randomized block, just like in Subsection 3.1. Observe that, for all $k \in \{1, \ldots, K\}$, we have : $\mathbb{E}[\bar{U}_k(h) \mid \mathcal{S}_n] = U_n(h)$. The *Median of Randomized U-statistics* estimator of $\theta(h)$ is then defined as

$$\bar{\theta}_{\text{MoRU}}(h) = \text{median}(\bar{U}_1(h), \ldots, \bar{U}_K(h)). \qquad (13)$$

The following proposition establishes the accuracy of the estimator (13), while emphasizing the advantages of the greater flexibility it offers to choose B and K.

Proposition 3. *For any $\tau \in (0, 1/2)$, for any $\delta \in [2e^{-8\tau^2 n/9}, 1)$, choosing $K = \log(2/\delta)/(2(1/2-\tau)^2)$ and $B = 8\tau^2 n/(9\log(2/\delta))$, it holds w.p.a.l. $1-\delta$:*

$$\left|\bar{\theta}_{MoRU}(h) - \theta(h)\right| \leq \sqrt{\frac{C_1(\tau)\log\frac{2}{\delta}}{n} + \frac{C_2(\tau)\log^2(\frac{2}{\delta})}{n\left(8n - 9\log\frac{2}{\delta}\right)}},$$

with $C_1(\tau) = 27\sigma_1^2(h)/(2\tau^3)$ and $C_2 = 243\sigma_2^2(h)/(4\tau^3)$.

Remark 2. *Observe that Proposition 2's constants (and bound) can be recovered asymptotically by letting $\tau \to 1/2$.*

3.3 MoU-based Pairwise Learning

We now describe a version of the tournament method tailored to *pairwise learning*. Let $\mathcal{X}$ be a measurable space, $\mathcal{F} \subset \mathbb{R}^{\mathcal{X}\times\mathcal{X}}$ a class of decision rules, and $\ell : \mathcal{F} \times \mathcal{X}^2 \to \mathbb{R}_+$ a given loss function. The goal pursued here is to learn from $2n$ i.i.d. variables $X_1, \ldots, X_{2n}$ distributed as a generic r.v. X valued in $\mathcal{X}$ a minimizer of the risk $\mathcal{R}(f) = \mathbb{E}[\ell(f, (X, X'))]$, where X' denotes an independent copy of X. In order to benefit from the standard tournament setting, we introduce the following notation : for every $f \in \mathcal{F}$, let $H_f(X, X') = \sqrt{\ell(f, (X, X'))}$ the kernel that maps every pair (X, X') to its (square root) loss through f. Let $\mathcal{H}_\mathcal{F} = \{H_f : f \in \mathcal{F}\}$. It is easy to see that for all $f \in \mathcal{F}$, $\mathcal{R}(f) = \|H_f\|^2_{L_2(\mu)}$, and that if f^* and H^* denote respectively the $\mathcal{R}$ and L_2 minimizers over $\mathcal{F}$ and $\mathcal{H}_\mathcal{F}$, then $H^* = H_{f^*}$. First, the dataset is split into 2 subsamples $\mathcal{S}$ and $\mathcal{S}'$ each of size n. As in [LM16], a distance oracle is used to allow matches to take place. Namely, for any $f, g \in \mathcal{F}^2$, let $\Phi_\mathcal{S}(f, g)$ be a MoU estimate of $\|H_f - H_g\|_{L_1}$ built on $\mathcal{S}$. If $\mathcal{B}_1, \ldots, \mathcal{B}_K$ is a partition of $\mathcal{S}$, it reads :

$$\Phi_\mathcal{S}(f, g) = \text{med}\left(\hat{U}_1(|H_f - H_g|), \ldots, \hat{U}_K(|H_f - H_g|)\right).$$

If $\Phi_\mathcal{S}(f, g)$ is greater than βr, for β and r to be specified later, a match between f and g is allowed. A MoRU estimate of $\|H_f - H_g\|_{L_1}$ could also have been used instead of a MoU. The idea underlying the pairwise tournament is the same than that of the standard one : with high probability $\Phi_\mathcal{S}(f, g)$ is a good estimate of $\|H_f - H_g\|_{L_2}$, so that only distant candidates are allowed to confront. And if H_{f^*} is one of these two candidates, it should hopefully win its match against a distant challenger. The nature of these matches is to be specified now. For any $f, g \in \mathcal{F}^2$, let $\Psi_{\mathcal{S}'}(f, g)$ denote a MoU estimate of $\mathbb{E}[H_f^2 - H_g^2]$ built on $\mathcal{S}'$. With $\mathcal{B}'_1, \ldots, \mathcal{B}'_{K'}$ a partition of $\mathcal{S}'$, $\Psi_{\mathcal{S}'}(f, g)$ reads

$$\Psi_{\mathcal{S}'}(f, g) = \text{med}\left(\hat{U}_1(H_f^2 - H_g^2), \ldots, \hat{U}_{K'}(H_f^2 - H_g^2)\right).$$

f is declared winner of the match if $\Psi_{\mathcal{S}'}(f, g) \leq 0$, *i.e.* if $\sum_{i<j} \ell(f, (X_i, X_j))$ is lower than $\sum_{i<j} \ell(g, (X_i, X_j))$ on more than half of the blocks. A candidate function $\hat{f}$ that has not lost a single match it has been allowed to participate in presents good generalization properties under mild assumptions, as revealed by the following theorem.

Theorem 1. *Let $q > 2$, $L > 1$. There exist constants c, c_0, c_1 and c_2 that depend only on q and L for which the following holds. Let $\mathcal{F}$ be a class of functions, and ℓ a loss function such that $\mathcal{H}_\mathcal{F}$ is locally compact. Assume that $\forall H_f \in span(\mathcal{H}_\mathcal{F}), \|H_f\|_{L_q} \leq L\|H_f\|_{L_2}$. Let $r^* = r^*(f^*, c_1, c_2)$ defined analogously as in [LM16], a constant that only depends on the geometry of $\mathcal{H}_\mathcal{F}$ around H_{f^*}, and $r \geq 2r^*$. The function $\hat{f} \in \mathcal{F}$ returned by the previously described tournament procedure verify with probability at least $1 - \exp(c_0 n \min\{1, r^2\})$,*

$$\mathcal{R}(\hat{f}) - \mathcal{R}(f^*) \leq cr.$$

Démonstration. The proof is analogous to that of Theorem 2.11 in [LM16], instead that arguments for U-statistics are used.

3.4 Discussion - Further Extensions

The computation of the U-statistic (2) is expensive in the sense that it involves the summation of $\mathcal{O}(n^2)$ terms. The concept of *incomplete U-statistic*, see [Blo76], precisely permits to address this computational issue and achieve a trade-off between scalability and variance reduction. In one of its simplest forms, it consists in selecting a subsample of size $M \geq 1$ by sampling with replacement in the set of all pairs of observations that can be formed from the original sample. Setting $\Lambda = \{(i,j) : 1 \leq i < j \leq n\}$, and denoting by $\{(i_1, j_1), \ldots, (i_M, j_M)\} \subset \Lambda$ the subsample drawn by Monte-Carlo, the incomplete version of the U-statistic (2) is : $\widetilde{U}_M(h) = (1/M)\sum_{m \leq M} h(X_{i_m}, X_{j_m})$. As can be easily shown, it is an unbiased estimator of θ and has variance :

$$\mathrm{Var}\left(\widetilde{U}_M(h)\right) = \left(1 - \frac{1}{M}\right)\mathrm{Var}(U_n(h)) + \frac{\sigma^2(h)}{M}.$$

The difference between its variance and that of (2) vanishes as M increases. In contrast, when $M \leq \#\Lambda = n(n-1)/2$, the variance of a complete U-statistic based on a subsample of size $\lfloor\sqrt{M}\rfloor$, and thus on $\mathcal{O}(M)$ pairs just like $\widetilde{U}_M(h)$, is of order $\mathcal{O}(1/\sqrt{M})$. Minimization of incomplete U-statistics has been investigated in [CCB16] from the perspective of scalable statistical learning. Hence, rather than sampling first observations and forming next pairs from data blocks in order to compute a collection of *complete U-statistics*, which the median is subsequently taken of, one could sample directly pairs of observations, compute alternatively estimates of drastically reduced variance and output a *Median of Incomplete U-statistics*. However, one faces significant difficulties when trying to analyze theoretically the performance of such a variant, see also Remark 1.

4 Numerical Experiments

Here we display numerical results supporting the relevance of the MoM variants analyzed in the previous sections.

MoRM experiments. Considering inference of the expectation of four specified distributions (Gaussian, Student, Log-normal and Pareto), based on a sample of size $n = 1000$, seven estimators are compared below : standard MoM, and six MoRM estimators, related to different sampling schemes (SRSWoR, Monte-Carlo) or different values of the tuning parameter τ. Results are obtained through 5000 replications of the estimation procedures (see Table 1). Beyond the quadratic risk, accuracy of the estimators are assessed by means of deviation probabilities (see Figure 1), *i.e.* empirical quantiles for a geometrical grid of confidence levels δ. As highlighted above, $\tau = 1/6$ leads to (approximately) the same number of blocks as in the MoM procedure. However, MoRM usually select blocks of cardinality lower than n/K, so that the MoRM estimator with $\tau = 1/6$ uses less examples than MoM. Proposition 1 exhibits a higher constant for MoRM in that case, and it is confirmed empirically here. The choice $\tau = 3/10$ guarantees that the number of MoRM blocks multiplied by their cardinality is equal to n. This way, MoRM uses as much samples as MoM. Nevertheless, the increased variability leads to a slightly lower performance in this case. Finally, $\tau = 9/20$ is chosen to be closer to $1/2$, as suggested by (8). In this setting, the two constant factors are (almost) equal, and MoRM even empirically shows a systematic improvement compared to MoM. Note that the quantile curves should be decreasing. However, the estimators being δ-dependent, different experiments are run for each value of δ, and the rare little increases are due to this random effect.

Mo(R)U experiments. In these experiments assessing empirically the performance of the Mo(R)U methods, the parameter of interest is the variance (*i.e.* $h(x,y) = (x-y)^2/2$) of the four laws used above. Again, estimators are assessed through their quadratic risk (Table 2) and empirical quantiles (Figure 2).

5 Conclusion

In this paper, various extensions of the Medians-of-Means methodology, which tournament-based statistical learning techniques recently proposed in the literature to handle heavy-tailed situations rely on, have been investigated at length. First, confidence bounds showing that accuracy can be fully preserved when data blocks are built through SRSWoR schemes rather than simple segmentation, giving more flexibility to the approach regarding the number of blocks and their size. Second, its application to estimation of pairwise expectations (*i.e.* Medians-of-U-statistics) is studied in a valid theoretical framework, paving the way for the design of robust pairwise statistical learning techniques in clustering, ranking or similarity learning.

Table 1 – Quadratic Risks for the Mean Estimation, $\delta = 0.001$

	Normal $(0,1)$	Student (3)	Log-normal $(0,1)$	Pareto (3)
MoM	0.00149 ± 0.00218	0.00410 ± 0.00584	0.00697 ± 0.00948	$\mathbf{1.02036} \pm \mathbf{0.06115}$
$\text{MoRM}_{1/6,\ \text{SWoR}}$	0.01366 ± 0.01888	0.02947 ± 0.04452	0.06210 ± 0.07876	1.12256 ± 0.14970
$\text{MoRM}_{3/10,\ \text{SWoR}}$	0.00255 ± 0.00361	0.00602 ± 0.00868	0.01241 ± 0.01610	1.05458 ± 0.07041
$\text{MoRM}_{9/20,\ \text{SWoR}}$	$\mathbf{0.00105} \pm \mathbf{0.00148}$	$\mathbf{0.00264} \pm \mathbf{0.00372}$	$\mathbf{0.00497} \pm \mathbf{0.00668}$	1.02802 ± 0.04903

Table 2 – Quadratic Risks for the Variance Estimation, $\delta = 0.001$

	Normal $(0,1)$	Student (3)	Log-normal $(0,1)$	Pareto (3)
$\text{MoU}_{1/2\,;\,1/2}$	0.00409 ± 0.00579	1.72618 ± 28.3563	2.61283 ± 23.5001	1.35748 ± 36.7998
$\text{MoU}_{\text{Partition}}$	$\mathbf{0.00324} \pm \mathbf{0.00448}$	$\mathbf{0.38242} \pm \mathbf{0.31934}$	$\mathbf{1.62258} \pm \mathbf{1.41839}$	$\mathbf{0.09300} \pm \mathbf{0.05650}$
$\text{MoRU}_{\text{SWoR}}$	0.00504 ± 0.00705	0.51202 ± 3.88291	2.01399 ± 4.85311	0.09703 ± 0.07116

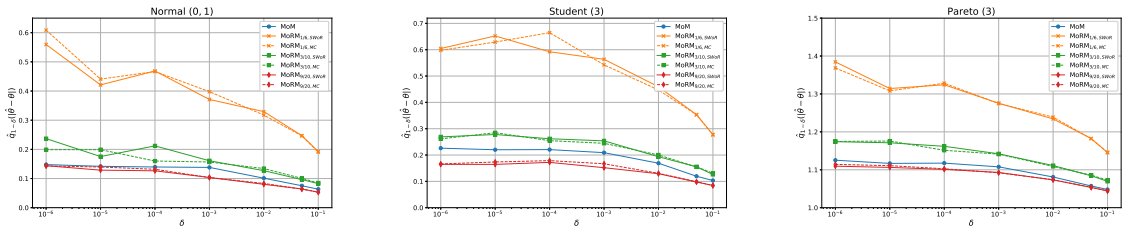


Figure 1 – Empirical Quantiles for the Different Mean Estimators on 3 Laws

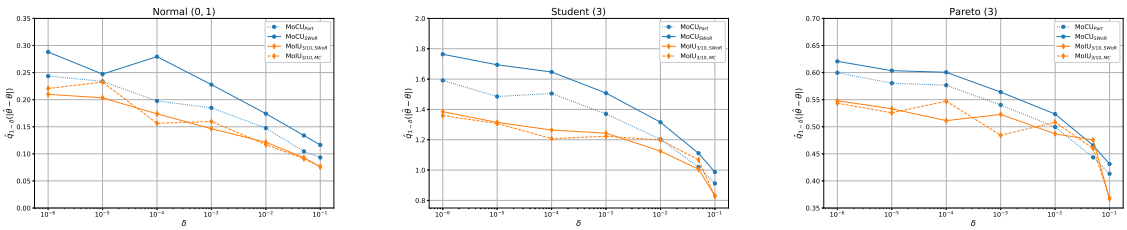


Figure 2 – Empirical Quantiles for the Different Variance Estimators on 3 Laws

Technical Proofs

Proof of Proposition 1

Let $\varepsilon > 0$. First observe that

$$\left\{\left|\bar{\theta}_{\text{MoRM}} - \theta\right| > \varepsilon\right\} \subset \left\{\sum_{k=1}^{K} I_\varepsilon(\mathcal{B}_{\boldsymbol{\epsilon}_k}) \geq K/2\right\},$$

where $I_\varepsilon(\mathcal{B}_{\boldsymbol{\epsilon}_k}) = \mathbb{I}\{\left|\bar{\theta}_k - \theta\right| > \varepsilon\}$ for $k \leq K$. To benefit from the independence of the blocks given $\mathcal{S}_n$, we condition upon $\mathcal{S}_n$ and consider the $\boldsymbol{\epsilon}_k$'s variability :

$$\mathbb{P}\left\{\left|\bar{\theta}_{\text{MoRM}} - \theta\right| > \varepsilon \mid \mathcal{S}_n\right\} \leq \mathbb{P}\left\{\sum_{k=1}^{K} \frac{I_\varepsilon(\mathcal{B}_{\boldsymbol{\epsilon}_k})}{K} \geq \frac{1}{2}\middle|\mathcal{S}_n\right\}.$$

Now, the average $(1/K)\sum_{k=1}^{K} I_\varepsilon(\mathcal{B}_{\boldsymbol{\epsilon}_k})$ can be viewed as an approximation of the U-statistic of degree B (refer to [Lee90]), its conditional expectation given $\mathcal{S}_n$ being

$$U_n^\varepsilon = \frac{1}{\binom{n}{B}} \sum_{\boldsymbol{\epsilon} \in \Lambda(n,B)} I_\varepsilon(\mathcal{B}_{\boldsymbol{\epsilon}}).$$

Denoting by $p^\varepsilon = \mathbb{E}[U_n^\varepsilon] = \mathbb{P}\{|\bar{\theta}_1 - \theta| > \varepsilon\}$ the expectation of the $I_\varepsilon(\mathcal{B}_{\boldsymbol{\epsilon}_k})$'s, we have $\forall \tau \in (0,1/2)$:

$$\mathbb{P}\{\left|\bar{\theta}_{\mathrm{MoRM}} - \theta\right| > \varepsilon\} \leq \mathbb{P}_{\mathcal{S}_n}\{U_n^\varepsilon - p^\varepsilon \geq \tau - p^\varepsilon\} \quad (14)$$
$$+ \mathbb{E}_{\mathcal{S}_n}\left[\mathbb{P}_{\boldsymbol{\epsilon}}\left\{\frac{1}{K}\sum_{k=1}^{K} I_\varepsilon(\mathcal{B}_{\boldsymbol{\epsilon}_k}) - U_n^\varepsilon \geq \frac{1}{2} - \tau \middle| \mathcal{S}_n\right\}\right].$$

By virtue of Hoeffding inequality for i.i.d. averages (see [Hoe63]) conditioned upon $\mathcal{S}_n$, we have : $\forall t > 0$,

$$\mathbb{P}_{\boldsymbol{\epsilon}}\left\{\frac{1}{K}\sum_{k=1}^{K} I_\varepsilon(\mathcal{B}_{\boldsymbol{\epsilon}_k}) - U_n^\varepsilon \geq t \middle| \mathcal{S}_n\right\} \leq \exp\left(-2Kt^2\right). \quad (15)$$

In addition, the version of Hoeffding inequality for U-statistics (*cf* [Hoe63], see also Theorem A in Chapter 5 of [Ser80]) yields : $\forall t > 0$,

$$\mathbb{P}_{\mathcal{S}_n}\{U_n^\varepsilon - p^\varepsilon \geq t\} \leq \exp\left(-2nt^2/B\right). \quad (16)$$

One may also show that $p^\varepsilon \leq \frac{\sigma^2}{B\epsilon^2}$. Hence, it follows :

$$\mathbb{P}_{\mathcal{S}_n}\{U_n^\varepsilon - p^\varepsilon \geq \tau - p^\varepsilon\} \leq \mathbb{P}_{\mathcal{S}_n}\left\{U_n^\varepsilon - p^\varepsilon \geq \tau - \frac{\sigma^2}{B\varepsilon^2}\right\}. \quad (17)$$

Combining equations (14), (15), (16) and (17), the deviation probability of $\bar{\theta}_{\mathrm{MoRM}}$ can be bounded by

$$\exp\left(-2\frac{n}{B}\left(\tau - \frac{\sigma^2}{B\varepsilon^2}\right)^2\right) + \exp\left(-2K\left(\frac{1}{2} - \tau\right)^2\right). \quad (18)$$

Choosing $K = \lceil \log(2/\delta)/(2(1/2-\tau)^2) \rceil$ and $B = \lfloor 8\tau^2 n/(9\log(2/\delta)) \rfloor$ leads to the desired result. □

Proof of Proposition 2

The data blocks are built here by partitioning the dataset into $K \leq n$ subsamples of size $B = \lfloor n/K \rfloor$. Set $I_{\varepsilon,k} = \mathbb{I}\{|\hat{U}_k(h) - \theta(h)| > \varepsilon\}$ for $k \leq K$. Again, observe that $\mathbb{P}\left\{\left|\hat{\theta}_{\mathrm{MoU}}(h)(h) - \theta(h)\right| > \varepsilon\right\}$ is lower than

$$\mathbb{P}\left\{\frac{1}{K}\sum_{k=1}^{K} I_{\varepsilon,k} - q^\varepsilon \geq \frac{1}{2} - q^\varepsilon\right\},$$

where $q^\varepsilon = \mathbb{E}[I_{\varepsilon,1}] = \mathbb{P}\{|\hat{U}_1(h) - \theta(h)| > \varepsilon\}$. By virtue of Chebyshev's inequality and equation (3) :

$$q^\varepsilon \leq \frac{\mathrm{Var}\left(\hat{U}_1(h)\right)}{\varepsilon^2} = \frac{1}{\varepsilon^2}\left(\frac{4\sigma_1^2(h)}{B} + \frac{2\sigma_2^2(h)}{B(B-1)}\right).$$

By Hoeffding, the deviation probability is bounded by

$$\exp\left(-2K\left(\frac{1}{2} - \left(\frac{4\sigma_1^2(h)}{B\varepsilon^2} + \frac{2\sigma_2^2(h)}{B(B-1)\varepsilon^2}\right)\right)^2\right).$$

Choosing $K = \log(1/\delta)/(2(1/2-\lambda)^2)$, $\lambda \in (0,1/2)$ gives :

$$\varepsilon = \sqrt{C_1(\lambda)\frac{\log(\frac{1}{\delta})}{n} + C_2(\lambda)\frac{\log^2(\frac{1}{\delta})}{n\left[2(\frac{1}{2}-\lambda)^2 n - \log(\frac{1}{\delta})\right]}},$$

with $C_1(\lambda) = 2\sigma_1^2(h)/(\lambda(\frac{1}{2}-\lambda)^2)$ and $C_2(\lambda) = \sigma_2^2(h)/(\lambda(1/2-\lambda)^2)$. The optimal constant for the first and leading term is attained for $\lambda = 1/6$, which corresponds to $K = (9/2)\log(1/\delta)$ and gives :

$$\varepsilon = \sqrt{C_1\frac{\log(\frac{1}{\delta})}{n} + C_2\frac{\log^2(\frac{1}{\delta})}{n\left(2n - 9\log(\frac{1}{\delta})\right)}},$$

with $C_1 = 108\sigma_1^2(h)$ and $C_2 = 486\sigma_2^2(h)$. □

Proof of Proposition 3

Here we consider the situation where the estimator is the median of K randomized U-statistics, computed from data blocks built by means of independent SRSWoR schemes. We set $\mathcal{I}_\varepsilon(\boldsymbol{\epsilon}_k) = \mathbb{I}\left\{|\bar{U}_k(h) - \theta(h)| > \varepsilon\right\}$. For all $\tau \in (0,1/2)$, we have :

$$\mathbb{P}\left\{\left|\bar{\theta}_{\mathrm{MoRU}}(h) - \theta(h)\right| > \varepsilon\right\} \leq \mathbb{P}\left\{\frac{1}{K}\sum_{k=1}^{K}\mathcal{I}_\varepsilon(\boldsymbol{\epsilon}_k) \geq \frac{1}{2}\right\}$$
$$\leq \mathbb{E}\left[\mathbb{P}\left\{\frac{1}{K}\sum_{k=1}^{K}\mathcal{I}_\varepsilon(\boldsymbol{\epsilon}_k) - W_n^\varepsilon \geq \frac{1}{2} - \tau \mid \mathcal{S}_n\right\}\right]$$
$$+ \mathbb{P}\left\{W_n^\varepsilon - \bar{q}^\varepsilon \geq \tau - \bar{q}^\varepsilon\right\}, \quad (19)$$

where

$$W_n^\varepsilon = \mathbb{E}\left[\mathcal{I}_\varepsilon(\boldsymbol{\epsilon}_1) \mid \mathcal{S}_n\right] = \frac{1}{\binom{n}{B}}\sum_{\epsilon \in \Lambda_{n,B}} \mathcal{I}_\varepsilon(\epsilon),$$
$$\bar{q}^\varepsilon = \mathbb{E}[\mathcal{I}_\varepsilon(\boldsymbol{\epsilon}_1)] = \mathbb{P}\{|\bar{U}_1(h) - \theta(h)| > \varepsilon\}.$$

The conditional expectation W_n^ε, which has mean $\bar{q}^\varepsilon$, is a U-statistic of degree B, so that the variant of Hoeffding inequality proved in [Ser80] (see Theorem A in Chapter 5 therein) yields :

$$\mathbb{P}\left\{W_n^\varepsilon - \bar{q}^\varepsilon \geq \tau - \bar{q}^\varepsilon\right\} \leq \exp\left(-2\frac{n}{B}\left(\frac{1}{2} - \bar{q}^\varepsilon\right)^2\right). \quad (20)$$

One may also show that

$$\bar{q}^\varepsilon \leq \frac{1}{\varepsilon^2}\left(\frac{4\sigma_1^2(h)}{B} + \frac{2\sigma_2^2(h)}{B(B-1)}\right). \quad (21)$$

Combining (19), Hoeffding's inequality conditioned on the data, (20) and (21), with $K = \log(2/\delta)/(2(1/2-\tau)^2)$ and $B = 8\tau^2 n/(9\log(2/\delta))$ leads to the desired bound. □

Références

[AC11] Jean-Yves Audibert and Olivier Catoni. Robust linear least squares regression. *The Annals of Statistics*, 39(5) :2766–2794, 2011.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1) :137–147, 1999.

[BCBL13] Sébastien Bubeck, Nicolo Cesa-Bianchi, and Gábor Lugosi. Bandits with heavy tail. *IEEE Transactions on Information Theory*, 59(11) :7711–7717, 2013.

[BHS13] A. Bellet, A. Habrard, and M. Sebban. A Survey on Metric Learning for Feature Vectors and Structured Data. *ArXiv e-prints*, June 2013.

[BJL+15] Christian Brownlees, Emilien Joly, Gábor Lugosi, et al. Empirical risk minimization for heavy-tailed losses. *The Annals of Statistics*, 43(6) :2507–2536, 2015.

[Blo76] G. Blom. Some properties of incomplete U-statistics. *Biometrika*, 63(3) :573–580, 1976.

[Cat12] Olivier Catoni. Challenging the empirical mean and empirical variance : a deviation study. In *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, volume 48, pages 1148–1185. Institut Henri Poincaré, 2012.

[CCB16] S. Clémençon, I. Colin, and A. Bellet. Scaling-up Empirical Risk Minimization : Optimization of Incomplete U-statistics. *Journal of Machine Learning Research*, 17 :1–36, 2016.

[Clé14] Stéphan Clémençon. A statistical view of clustering performance through the theory of u-processes. *Journal of Multivariate Analysis*, 124 :42–56, 2014.

[CLV05] S. Clémençon, G. Lugosi, and N. Vayatis. Ranking and scoring using empirical risk minimization. In *Proceedings of COLT*, 2005.

[CLV08] S. Clémençon, G. Lugosi, and N. Vayatis. Ranking and empirical risk minimization of U-statistics. *The Annals of Statistics*, 36(2) :844–874, 2008.

[DLL+16] Luc Devroye, Matthieu Lerasle, Gabor Lugosi, Roberto I Oliveira, et al. Sub-gaussian mean estimators. *The Annals of Statistics*, 44(6) :2695–2725, 2016.

[Hoe48] W. Hoeffding. A class of statistics with asymptotically normal distribution. *Ann. Math. Stat.*, 19 :293–325, 1948.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301) :13–30, 1963.

[HS16] Daniel Hsu and Sivan Sabato. Loss minimization and parameter estimation with heavy tails. *The Journal of Machine Learning Research*, 17(1) :543–582, 2016.

[JVV86] Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43 :169–188, 1986.

[Lee90] A. J. Lee. *U-statistics : Theory and practice*. Marcel Dekker, Inc., New York, 1990.

[LM16] Gabor Lugosi and Shahar Mendelson. Risk minimization by median-of-means tournaments. *arXiv preprint arXiv :1608.00757*, 2016.

[LM17] Gábor Lugosi and Shahar Mendelson. Subgaussian estimators of the mean of a random vector. *arXiv preprint arXiv :1702.00482*, 2017.

[LO11] Matthieu Lerasle and Roberto I Oliveira. Robust empirical mean estimators. *arXiv preprint arXiv :1112.3914*, 2011.

[M+15] Stanislav Minsker et al. Geometric Median and Robust Estimation in Banach Spaces. *Bernoulli*, 21(4) :2308–2335, 2015.

[NY83] Arkadii Semenovich Nemirovsky and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

[Ser80] R.J. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons, 1980.

[VCB18] R. Vogel, S. Clémençon, and A. Bellet. A Probabilistic Theory of Supervised Similarity Learning : Pairwise Bipartite Ranking and Pointwise ROC Curve Optimization. In *International Conference in Machine Learning*, 2018.

[VdV00] A.W. Van der Vaart. *Asymptotic Statistics*. Cambridge university press, 2000.

Input Addition in Reinforcement: Towards Learning with Structural Changes

Iago Bonnici[1], Abdelkader Gouaïch[1], and Fabien Michel[1]

[1]LIRMM, Université de Montpellier, CNRS, Montpellier

May 29, 2019

Abstract

Reinforcement Learning (RL) involves an artificial agent feeding from input data streams and producing output streams in order to maintain high rewards values. This is useful as RL can represent any environment with an input/output stream signature, like robots sensors/actuators, or computer programs and their relations to user. Natural situations involve changes in the environment, which are typically tackled under the assumption that the agent stream signature remains the same. However, the agent signature may also vary *e.g.* when an actuator crashes or a new sensor is plugged in. In this paper, we get interested in this unusual situation that we call "protean" learning (PL). In particular, we assess how the learner can rely on its past shapes and experience to keep improving among signature changes without needing to restart the learning from scratch on each change. We first present a formalization of the PL problem. Then, we address the first elementary signature change: *input addition*, with Recurrent Neural Networks (RNNs) in an idealized PL situation. We find that it is possible to benefit from prior learning in RNNs even if the past controller agent signature has less inputs than the current one. The use of PL thus remains encouraged. Investigating output addition, input/output removal and translating these results to generic PL will be part of future works.

keywords: reinforcement learning, transfer learning, recurrent networks, structural changes

1 Introduction

1.1 Reinforcement Learning

In heuristic, weak AI, Artificial Agents (AAs) are mostly used as *search* tools. They are assigned a task that their user may not know how to tackle, and their goal is to search a wide space of possible behaviours until it they find one that makes them solve the task, at least approximately. In Supervised Learning situations (SL) the AA can feed on examples realisations of the target behaviour. *I.e.* a series of correct mappings $\mathsf{input} \mapsto \mathsf{output}$ called a *training set*, that can guide it towards acceptable approximations of the target. In Reinforcement Learning situations (RL) however, no such mappings are available. Instead, the agent must successively try various behaviours, only guided with a continuously fed *reward* value, designed by user, used as clue regarding whether or not it is doing well [SB18, HHCB10]. In this context, a *behaviour* is a function continuously computing the next "actions" to undergo (the agent *outputs*) based on current perceived "state" of the system (the agent *inputs*). As this design is simple and natural, it is helpful in a variety of situation like robots animation (*e.g.* believable behaviours [LHABL04, TB10, PMJ+16]) or progressive adaptation of software parameters to their user (*e.g.* language habits, adaptive games [HM00, Che07]). RL benefits a lot from the development of tools like neural networks, in particular Recurrent Neural Networks (RNNs) that are well suited for sequential processing [Elm90, Sch15].

1.2 The protean situation

Ideal RL agents can adapt changing conditions, while still solving the task at hand. In this paper, we consider that not only the agent *environment* can change, but also the agent *shape* as it may undergo structural transformations during the course of its activity.

For instance, consider a RL agent embed into a sticky roverbot, whose task is to follow its user anywhere. Feeding from its sensors inputs, and starting from an initial, trivial behaviour where it does not move at all, this agent has to progressively learn how to coordinate its actions until it is able to follow its target. However, the successful state where it is able to solve this task on a wooden floor means

not that it will be able to follow the user later on a rocky ground (*environmental change*), when its rear camera will break (*input deletion*), when its right caterpillar eventually gets jammed (*output deletion*), when a new engine will be added to compensate (*output addition*), or when a new infrared sensor will be plugged in (*input addition*).

By explicitly taking this kind of structural changes into account, we expect extending RL to a broader learning situation which we refer to as Protean Learning (PL). PL agents should be able to keep learning no matter changes in their list of available inputs and outputs, which we refer to as the agent's *signature*. Suppose that a RL agent has been trained to optimize its reward feedbacks under the first signature Δ_0. Our question is: How can this agent be used later under other signatures like Δ_1, Δ_2, *etc.*? Is it more interesting to restart the training from scratch on each signature change, or is it possible to benefit from previous experience even though the signature is not the same?

We trial this question by exploring and evaluating a learning solution which capitalizes on past experience, even when the signature partially changes. After a succinct overview of related works (section 1.3), we offer a formalization of PL (section 2.2) and a description of the experimental setting addressing a first change: *input addition* (section 2.3) Section 3 exposes and discusses first results before we conclude.

1.3 Related work

Situations similar to PL are known in the domain of Transfer Learning (TL). In TL, the agent has already learned tasks called "source" tasks, and the challenge is to benefit from this previous "knowledge" while tackling a new "target" task. In other words, the TL agent is expected to generalize not only *within* tasks, but also *across* tasks [TS09, Laz12]. This domain is transversal to Machine Learning as it applies both to SL and RL. TL may be invoked in various situations: **(1)** The source training process has been successful but costly: one wishes to benefit from transfer to tackle a new target task more efficiently [TY03, TS09]. **(2)** The target task is challenging: one wishes to split it up into several easier source tasks, expecting that the overall TL process will be more efficient than direct tackling of the target [TS09, DGD$^+$16, FHC$^+$17]. **(3)** Several tasks must be learned at once: one wishes that TL occurs from the ones to the others, and speeds up the parallel process [TBC$^+$17, HM10]. **(4)** The task at hand will undergo future changes, but they are not known yet: one wishes to design an agent able to adapt these changes and benefit from transfer from one task to the next. PL is an instance of the latter situation.

PL is also related to Concept Drift (CD), a situation where the environment is assumed to undergo changes while the agent keeps learning [Tsy04, HA15]. It is also related to Continual Learning (CL) or "lifelong learning", an AI design where the agent keeps learning although its environment is changing and it regularly faces new challenges [Rin94, XZ18]. To our knowledge, even though the environment function is expected to change in TL, CD, CL, the signature of the agent is widely assumed to be fixed. The field of Domain Adaptation (DA) tackles input changes (heterogeneous DA [DXT12, HM10]) or output changes (open-set DA [BG17]) in SL and the first transfer situation **(1)**. Here, we focus on signature changes in RL and the last transfer situation **(4)**, and how transfer can be achieved in this case with the classical RNNs tools. This implies that the learning process be incremental [LHW18].

2 Material and Methods

2.1 Approach

The studying of PL must first be conducted with agents operating in simple environments. We propose that a 2D-simulated sticky roverbot like the one described in section 1.1 be used as a PL instance. This agent would feed, from its target location *e.g.* distance and angle as inputs. It would produce torque and thrust as outputs. And the inverse distance to its target would be continuously evaluated as a reward stream. Signature changes can easily be simulated in such a controlled situation. It will serve as an experimental setup in subsequent works.

As a first step, the abstract system that we call PL first needs be formalized (see section 2.2). Then, prior to testing PL in such toy applications, the basic relevance of PL has to be asserted. In section 2.3, we conduct a preliminary experiment involving RNNs in a idealized PL situation to assess the first operation among 6 elementary signature changes: input addition, input removal, output or feedback addition or removal. This experiment verifies that adding an input to the agent during the learning process does not dramatically alter learning, and that a protean learner performs better in this situation than a naive learner resuming from scratch on a signature change. Therefore, it encourages subsequent works towards the PL objective.

2.2 Formalization of PL

In this section, we sum up a formal specification of PL, based on detailed report [BG18]. PL is offered as a generalization of RL to non-fixed signature learning situations.

A RL agent is continuously fed with inputs so they can be considered data *streams*: values that change over time. It also feeds from feedback streams, and produces output

streams. Therefore, RL is viewed as a case of *stream processing*, where the learner continuously transforms the inputs streams into output streams *via* a process called its *behaviour*, with the objective that feedback streams values remain high. On the other hand, the environment continuously transforms the output streams into input streams and feedbacks, by strict application of the universe rules.

The agent *signature* is the number and the type of the streams it produces and feeds from. In other words, it is the collection of domains the various streams take their value in. The idea with this formalization is to make the signature a stream itself. As such, it may also change in time, which makes the agent *protean* and extends RL to PL.

We represent streams with plain functions of continuous time, like $h\colon \mathbb{R}^+ \to D$. They take their value in arbitrary domains D. Streams can be discretized in time with arbitrary precision $\varepsilon \in \mathbb{R}^{+*}$ by sequences ${}^{\varepsilon}h : \mathbb{N} \to D$ such that:

$$\forall t \in \mathbb{N},\ {}^{\varepsilon}h(t) = h(t\varepsilon) \tag{1}$$

Streams transform into each other. Viewed another way, a stream can be determined by another stream. We call a *determination function* f a function able to determine an outgoing stream k from an incoming stream h no matter the precision ε considered: $\forall \varepsilon \in \mathbb{R}^{+*},\ \forall t \in \mathbb{N}$,

$${}^{\varepsilon}k(t) = f_{\varepsilon}({}^{\varepsilon}h(t),\ {}^{\varepsilon}h(t-1),\ \ldots,\ {}^{\varepsilon}h(0)) \tag{2}$$

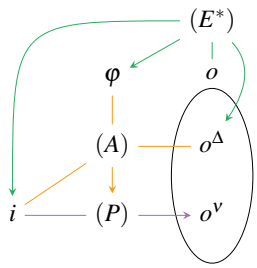
Stream determination has a *memory* in that current value of k may depend on past values of h, so it is not Markovian in general. However, it is always *causal* in that future values of k cannot be determined given only the current and past values of h. We shall use the following graphical alias for the determination relation (2):

$$h \text{ — } (f) \longrightarrow k \tag{3}$$

The symbol in parentheses represents the determination function, the symbol pointed by the arrow head is the consequence stream, and the symbol pointed by the line with no head is the cause stream. For instance, i — $(P) \longrightarrow o$ means that the inner agent process P feeds from input stream i to produce the output stream o in a causal, yet potentially non-Markovian way. Conversely, o — $(E) \longrightarrow i$ means that the environment E works the other way round.

A *multiple* stream h carries both a stream of domains h^{Δ} whose values are called *signatures*, and a stream of actual values h^{v} (see example Fig. 1 left). *Signatures* are a tuple of sets $(S_1,\ S_2,\ \ldots)$ and *values* are a tuple of elements from these sets $(v_1 \in S_1,\ v_2 \in S_2,\ \ldots)$. For instance, at $t = 0.9$, a sticky roverbot that is sensitive to both "user direction and ground speed" may receive $h^{\Delta}(0.9) = ([0,\ 2\pi[,\ \mathbb{R}^+)$ as a signature and $h^{v}(0.9) = \big(0.2\text{ rad},\ 15\text{ cm.s}^{-1}\big)$ as actual values. The particularity of PL, in contrast with RL, is that the signature stream is not constant. We call *transformation* of the PL agent any variation of h^{Δ} resulting in that the agent may later receive values with different domain signatures, thus the term *protean*. For instance, after its front camera has been broken, and a new battery sensor has been plugged in, the sticky agent may later receive "ground speed and battery level" $h^{\Delta}(1.1) = (\mathbb{R}^+,\ [\![1,\ 5]\!])$ and $h^{v}(1.1) = \big(31\text{ cm.s}^{-1},\ 4\big)$.

At the highest level, a PL learner agent can be represented by 3 multiple streams $(i,\ o,\ \varphi)$ and 1 stream of determining functions P with the following determination scheme:



(4)

The latter scheme can be considered a set of formal equations according to (3) and (2). Each colored arrow corresponds to one determination triplet, where:

- E represents the environment in which the agent is immersed. The * denotes that initial values of i, φ, o^{Δ} are determined by E. For the sticky roverbot, E would represent physics, the target user behavior, hardware and software environment all together. *I.e.* everything that is out of the decisional agent reach A.
- i represents the agent's *inputs* or *sensors*. For the sticky bot, i would inform the agent of user direction and distance. Their nature may change in time as the signature i^{Δ} evolves (*e.g.* on a sensor break or a sensor plug).
- o represents the agent's *outputs* or *actuators*. In our example, o controls the caterpillar engines. Their nature may also change in time as o^{Δ} evolves. Note that output values o^{v} are determined by the agent, but the output signature o^{Δ} is determined by the environment.
- φ represents the agent's *feedback*, *rewards* or *objectives*, a continuously fed evaluation of the actions it undertakes. For the sticky bot, φ would measure closeness to the target or the battery level. It also may change in nature as φ^{Δ} evolves.

The environment determines i, φ and o^{Δ}, so the agent cannot directly decide its input or feedback data, nor its output signature. On the inner side:

- P represents the agent current *behavior*, *policy* or *program*. It is an inner computational procedure that determines current output values based on current inputs and all past inputs. Concrete "decisions" of the agent, represented here as o^v, are produced by P. Note that P is a stream itself, so that it may evolve in time, and decisions taken by behavior $P(t)$ may not be taken later by behavior $P(t+\Delta_t)$.

- A is the abstract strategy of the agent, which is continuously adapting the behavior P based on environmental information. This is where the actual behaviour search is performed (see section 1.1). Abstract "decisions" of the agent, *i.e.* strategic choices represented here as P, are produced by A.

Only two objects are not depending on time in this system: the environment E and the inner agent strategy A. In the sense of Formal Grammars and Dynamical Systems, E and A embody the *rules* of the system, while its *initial state* is represented as the first production of E: $\big(i(0),\ \varphi(0),\ o^{\Delta}(0)\big)$.

The classical RL problem of "maximizing reward" [SB18] can be reformulated in PL as:

Given environment E and a corresponding stream of rewards φ with all domains being partially ordered, find an agent procedure A such that all values taken by stream φ^v are maximized.

A is considered an interesting learner if it can maximize the rewards for a whole family of environments, *i.e.* adapt many environments, no matter the variable nature of signatures streams i^{Δ}, o^{Δ} or φ^{Δ}.

2.3 Experiment

As a preliminary to the construction of generic PL agents, we design an experiment to assess viability of the first type of signature change: "input addition". To this end, we restrict ourselves to an ideal situation where the optimal behavior $\hat{P}$ is known from the experimenter. The agent A will be able to access a set of example optimal realizations $T = \{(i_k,\ \hat{o}_k)\}_{k\in[\![1,\ 1000]\!]}$ with every i_k — $(\hat{P}) \rightarrow \hat{o}_k$ (see section 2.2). This will constitute a *training set* so the agent search for an approximation of $\hat{P}$ can be solved with classical SL methods (see section 1.1). To simulate the signature change, we stop the SL procedure, change the dimension of each i_k, change the signature of A with a trivial transfer technique, and resume SL. Comparison is made with an agent that directly learns with the second signature. The less efficient the second agent compared to the first one, the more encouraging it is in favor the PL approach.

2.3.1 Inputs Generation

Idealized input i_k, are randomly generated 2-dimensional streams. They are either structured (smooth) on non-structured (noisy):

- In the noisy setting, each value is drawn from a uniform distribution so inputs have no structure:

$$i_k(t) \hookrightarrow \mathscr{U}\big([-1,\ 1]^2\big). \tag{5}$$

- In the smooth setting, each dimension of the stream is generated independently as a combination of sine waves, so that inputs are strongly autocorrelated. The generation procedure is described as follows (example Fig. 1 right top):

for each dimension:
  draw the number of sine waves \
    from a Poisson distribution $\mathscr{P}(500)$
  for each sine wave:
    draw pulsation ω from uniform $\mathscr{U}([0,\ 30])$
    draw phase φ from uniform $\mathscr{U}([0,\ 2\pi])$
    evaluate $\sin(\omega t+\varphi)$ for 120 values of t \
      evenly spaced between 0 and 2
  end for
  sum all sine waves into one stream sequence
  draw final amplitude a from uniform $\mathscr{U}([0,\ 1])$
  rescale the sequence so it has desired \
    amplitude: sequence $\leftarrow a \times \frac{\text{sequence}}{\max(|\text{sequence}|)}$
end for
join both sequences \
  into one input stream $i_k : [\![1,\ 120]\!] \rightarrow [-1,\ 1]^2$

2.3.2 Optimal Outputs Generation

Training outputs examples $\hat{o}_k$ are generated using idealized behavior $\hat{P}$. Depending on the experimental setting, two different formulae are used for $\hat{P}$ to address the effect of task complexity: one is instantaneous (nomem) while the other exhibits a small temporal memory (mem).

- In the nomem setting, $\hat{P}$ has no memory and performs a plain combination of the 2 input streams dimensions i_k^1 and i_k^2 that is affine in the atanh transformed space:

$$\hat{o}_k(t) = \tanh\big(p\,\text{atanh}\big(i_k^1(t)\big) + (1-p)\text{atanh}\big(i_k^2(t)\big) + c\big) \tag{6}$$

  With $c \in \mathbb{R},\ p \in [0,\ 1]$.

- In the mem setting, the processor is more complex because it exhibits a 1-step memory, so it is non-Markovian. It performs the same combination with the exception that $5 \times (i_k^d(t) - i_k^d(t-1))$ is used instead of

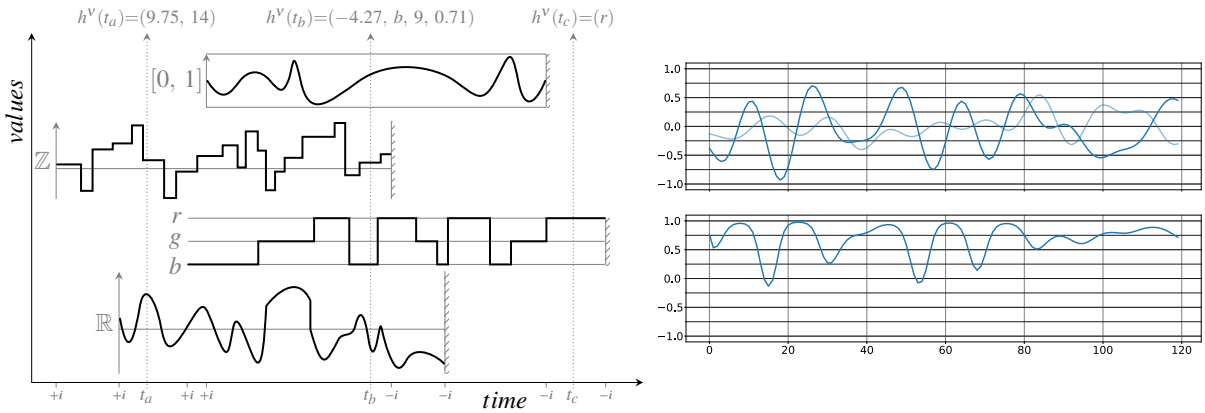


Figure 1: Left: Example multiple stream h. Each curve corresponds to the temporary presence of one domain in the signature stream h^{Δ}. If h represents an agent input streams, then this learner is initially sensitive to a measure valued in $\mathbb{Z}$. It later becomes also sensitive to a measure valued in $\mathbb{R}$, then in $[0,\ 1]$. All these sensitivities are eventually lost between t_b and t_c. However, the learner ends up sensitive to a nominal parameter in $\{r,\ g,\ b\}$. All beginning and end of sensitivities are marked as input addition $(+i)$ or input deletion $(-i)$ events. Right: Example synthetic streams for the preliminary experiment. Top: 2D input stream i_k (i_k^1 solid, i_k^2 faded). Bottom: 1D output stream $\hat{o}_k$. In this example, streams are smooth, the processing is mem, $p = 0.8$ (strong influence of i_k^1 on $\hat{o}_k$) and $c = 1$ (skewed output result).

$\operatorname{atanh}\left(i_k^d(t)\right)$ in (6). In other words, the derivative of input streams are used instead of their actual values.

The higher p, the more information contained in the first input stream dimension i_k^1, and not in the second i_k^2. Four values are tested for p to assess the effect of available information: 0 (first dimension irrelevant for solving the task), 0.2 (not sufficient for solving the task) 0.5 (a little more informative) and 0.8 (almost sufficient).

The further c from zero, the more skewed the resulting output stream $\hat{o}_k$ (Fig. 2 right bottom). Two values are tested for c to assess the effect of task bias: 0 (no bias) and 1 (strong bias).

2.3.3 Agent Structure

The agent approximates optimal behaviour $\hat{P}$ with its actual behaviour P. P takes the form of a Recurrent Neural Network (RNN) [Elm90, Sch15]. P produces the agent actual outputs according to i_k — $(P) \rightarrow o_k$. Two possible structures are tested for P:

- In the flat setting, 1 Gated Recurrent Unit cell (GRU) is used [CvG+14] with 1 internal state, used as network output. So P is Markovian.
- In the deep setting, 2 GRU cells are used as different network layers with 6 internal states, the last one being used as the network output. So P is more flexible and it may be non-Markovian.

2.3.4 Optimization

The learning procedure A processes training examples by batches of 10, and updates the weights parameters of P with a stochastic gradient descent following Adam update rule [KB14] (learning rate = 0.01). The loss function to optimize is the classical Mean Squared Error between the agent predictions and the expected results, calculated as $MSE(o,\ \hat{o})$ on each batch. Convergence is achieved using pytorch [PGC+17] for 1000 iterations.

2.3.5 Realization of Signature Change

Three convergences are achieved on each run (see Fig. 2):

1. One protean "first-form" agent A_{f_1} (blue trace) is constructed with a 1D-1D signature. Its parameters are randomly initialized, then it is trained against T but only feeds from the first dimension i_k^1 of input streams, staying blind to i_k^2.
2. One protean "second-form" agent A_{f_2} is constructed with a 2D-1D signature. Its initial parameters are copied from the latest parameters in A_{f_1}, except for the necessary additional parameters that are set to zero. This naive form of transfer is considered transfer of "low-level" knowledge in [TS09] with an obvious, available mapping from source to target task. A_{f_2} is then trained against the whole training set T (green trace), not ignoring the second dimension i_k^2 anymore.

3. One classical "direct" agent A_d is constructed with a 2D-1D signature. Its parameters are randomly initialized, then it is directly trained against the whole training set T (black trace).

The $\left(A_{f_1}, A_{f_2}\right)$ agent is the experimental model of a PL agent in the idealized learning situation, experiencing the elementary signature changes "input addition". 1000 replicates are run for each combination of experimental settings.

2.3.6 Measure of the Advantage

The advantage of PL compared to direct learning is estimated on learning curves l by a score calculated on each run as a mean *gain* (see Fig. 2) according to

$$\text{score} = \text{mean}\left(\log_2\left(\frac{l_{A_d}}{l_{A_{f_2}}}\right)\right) \tag{7}$$

score $= 1$ means that the second-form agent error is twice better as the direct agent on average. score $= 0$ means that they perform similarly on average.

3 Results

The scores obtained on each run are summarized in Fig. 3.

A linear model was fitted on the data to address relevance of observed variations. Prediction are represented as blue dots on Fig. 3. All interactions were considered between experimental settings considered as factors and one nested slope for parameter p considered numerical (degrees of freedom: 63698, residual stde: 0.7689). Convergence and analysis of the model have been achieved with R-Cran software [R C18]. All effects were different from zero with high significance *p-value* $\leqslant .001$ with the only exception of the nested slope flat:mem:smooth:c=0:p (*p-value* $= .8118$). We therefore consider that all trends among groups observable on Fig. 3 are relevant to discuss, except the latter slope that must be considered null.

However, convergence of the network and good approximation of $\hat{P}$ within the setting flat:mem is impossible because $\hat{P}$ (non-Markovian then) uses a memory information that neither A_{f_1}, A_{f_2} nor A_d (with a Markovian P then) can access. As a consequence, they all poorly approximate $\hat{P}$ so these score values must be considered with circumspection. Correctly interpreting these values requires further investigations under conditions where networks fail to converge, which is out of the scope of this preliminary paper.

As expected, the measured score is mostly positive on average no matter the experimental setting. This reflects the advantage of the second-form agent A_{f_2} compared to the direct agent A_d: *i.e.* performing the transfer with a PL approach is more beneficial than restarting the learning from scratch when the signature change occurs. This advantage needs to be qualified depending on the situation:

- *PL benefits from past information*: It is expected that, when the first dimension carries information that is still relevant to solving the current task, progress done by the preliminary agent A_{f_1} can be transferred to A_{f_2}. This is confirmed by the data because the higher p, the higher the score.

- *PL relies on processing skews*: It is expected that, the more skewed the expected result, the easier it is to transfer information about this skew to the next-form agent A_{f_2}. This is confirmed by the data because the higher c, the higher the score. When $c = 1$, the direct learner A_d has to learn both the skew and the formula $\hat{P}$, so it is disadvantaged against A_{f_2} that already approximates the skew correctly.

- *PL relies on input structure*: Similarly, it is expected that the more structured the data, the easier it is to transfer information about this structure to the next-form agent A_{f_2}. This is also confirmed by the data because score is better in smooth than in noisy settings.

- *PL reliance is robust to impossible tasks*: It is expected that transfer is useless if the initial input carries no relevant information at all with respect to solving the task. However, the measured scores are *still* positive on average in the noisy setting with $p = 0$ and $c = 0$. Our interpretation is that A_{f_1} may still learn a correct calibration/representation/preprocessing of the data in this case, even though it cannot solve the task, and that this knowledge can benefit to A_{f_2}.

- *PL advantage is lower with deep RNNs* (at least in nomem setting for we do not discuss flat:mem): Deeper RNNs are more flexible in the sense that there exists more combinations of their parameters that approximate the objective function $\hat{P}$ correctly. As a consequence, it is easier for A_d to find a path down the loss function and converge during the second PL phase, which makes A_{f_2} advantage less decisive at this stage. However, keep in mind that the proposed task is trivial and that most agents in the experiment are able to solve it within 1000 iterations. This results may not extend to non-trivial tasks where both shallow and deep networks struggle approaching $\hat{P}$. We expect that in this case, PL will rather be advantaged by deep networks because they would at least provide solutions where

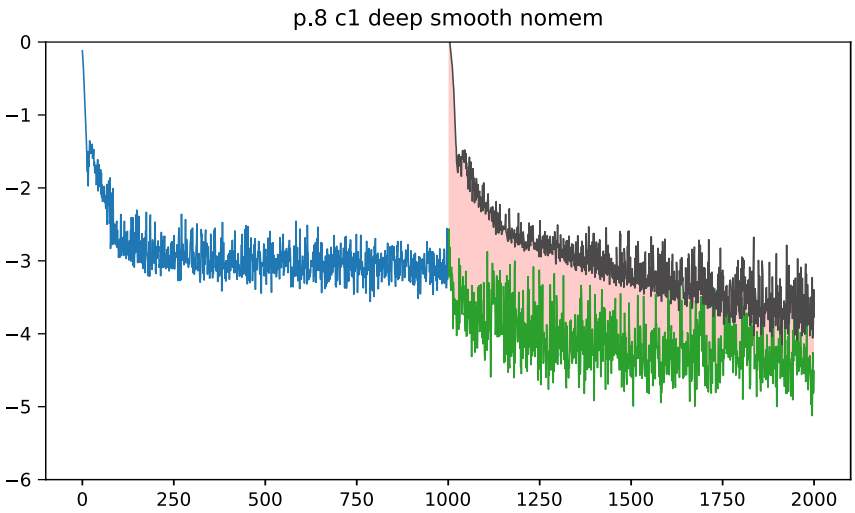


Figure 2: Example experimental learning curves l: evolution of the $\log_{10}(\text{MSE})$ (see section 2.3.5). The "first form" learner A_{f_1} ($l_{A_{f_1}}$ in blue) is trained to predict a transformation of a 2D stream $\left(i_k^1, i_k^2\right)$, but it can only access i_k^1 as an input so it has incomplete information. The "direct" learner A_d (l_{A_d} in black) is trained on the same task, but it can access the whole information so it performs better. The "second-form" learner A_{f_2} ($l_{A_{f_2}}$ in green) can also access the whole information so it has the same signature as A_d. But A_{f_2} initial parameters are copied from A_{f_1} instead of being randomly initialized. The gain score (in red ≈ 5.242) measures the benefit of PL (based on previous experience) compared to starting the learning from scratch on a signature change. The signature change event "input addition" occurs at $t = 1000$.

shallow networks would provide nothing. Assessing this will be part of future works.

As a summary, we observe that PL seems mostly beneficial after an "input addition" event, not only if past inputs are relevant ($p > 0$), but also if the learning data is somehow structured (c=1, smooth) and even if the first-form agent wast totally unable to solve the task ($p = 0$).

As TL is not yet backed by strong theoretical results [TS09], A_{f_1} can sometimes converge towards a "dead-end" direction during the first phase of the experiment, so it has to "unlearn" during the second phase and be disadvantaged against A_d. This phenomenon is known as *negative transfer* [TS09] and there is no guarantee against it. However, this is not the average behavior we observe here. We have thus demonstrated that transfer works as expected for the "input addition" signature change event, at least in this simplified situation, and that the protean approach of RL is still conceivable.

There exists several potential advantageous effects for the second-form agent A_{f_2} that benefits from transfer [TS09] (see Fig. 2). First, A_{f_2} initial loss may be lower than A_d, because A_{f_1} has already started convergence; a phenomenon known as *jumpstart* benefit. Second, A_{f_2} loss may decrease faster than A_d; so A_{f_2} is said to *learn faster*. Lastly, A_{f_2} final loss may be lower than A_d; so A_{f_2} is said to *learn better*. In our experiment, the *score* metric (7) is only an aggregated estimation of these 3 potential advantages. Therefore, we cannot distinguish all effects from each other. However, considering that the only difference between A_d and A_{f_2} is their initial RNN parameter values, we conjecture that the jumpstart effect is the major benefit of PL in this experiment.

The presented experiment is idealized because the agent can access a whole training set of optimal outputs $\hat{o}_k$, and it is directly guided by the loss gradient relative to its inner P RNN parameters. This is much unrealistic regarding RL. However, it is a valid instance of the generic PL procedure formalized in section 2.2. Extending these preliminary results to RL is therefore a matter of relaxing these ideal hypotheses and confront to Incremental Learning [LHW18] and the Credit Assignment problem [SB18]. As of today, we have no guarantee that the presented results will hold.

In subsequent works, consistently with our general approach (section 2.1), PL will be used again to address other types of signature changes like input removal, and output/feedback addition/removal. Then, actual PL agents will be constructed and trained in controlled environments like the 2D simulated sticky roverbot, before they can finally be applied to real-world problems: *e.g.* animating modular robots [ADDC14, DBME15] or powering adaptive game

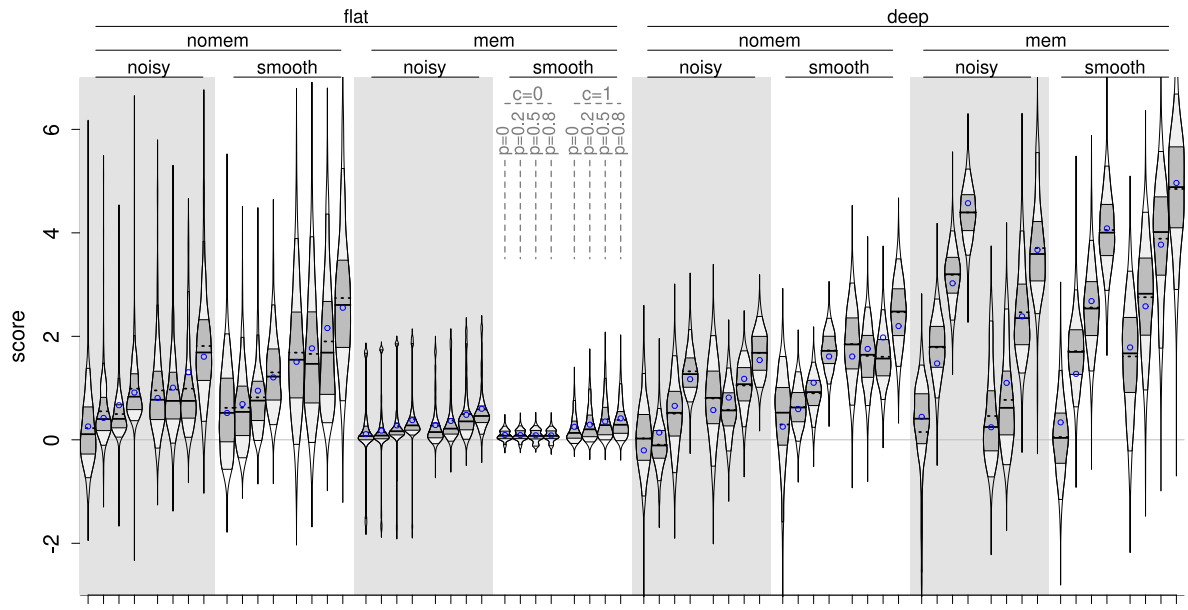


Figure 3: Violin plot: Comparison of scores in various PL situations differing by the RNN structure of P (flat or deep), the task complexity $\hat{P}$ (nomem or mem), the properties of input data i_k (smooth or noisy) and the relative informativeness of partial information (c, p). The benefit of PL differs depending on the situation, but is overall positive. Within each violin, the dashed line represents mean value for the group, the solid line represents median value, and the two gray areas represent 50% and 90% percentiles. Blue circles represent predictions of the linear model fitted on the data.

controllers [Fra14, HGC+15].

4 Conclusion

RL agents are expected to continuously adapt their environments. However, some real-world situations like hardware restructuring or software evolution challenge them with changes in their input/output signature. We generalize the idea of RL to a broader PL theoretical situation explicitly taking these structural changes into account. This enables a rigorous assessment of how this kind of learners could be developed. With a controlled, idealized experiment, we have shown that input addition can be addressed efficiently in PL at least with RNNs. Moreover, the benefits of PL in this situation are both easy to use (trivial low-level transfer technique) and robust to changes in the learning context (informativeness of partial inputs, task complexity, data structure). These first results are encouraging, suggesting that the protean approach can be both simple and robust, and that controlling complex, changing environments with PL is promising. The assessment of PL is still incomplete, since other basic operations must be tested: input removal and output/feedback addition/removal. Moreover, exporting these results from idealized PL to full-fledged, generic PL tasks still needs to be done.

Like any machine learning approach, PL is generic and can be applied in other domains provided a signature change in RL is identified. For instance, modular robotics could benefit from PL controllers. Long-term learners that cannot discard their past experience on a signature change also need to capitalize on it. And more generally, any bio-inspired agent that need to transform, split or merge, while keeping on learning, can benefit from a PL approach.

References

[ADDC14] Tarek Ababsa, Noureddine Djedi, Yves Duthen, and Sylvain Cussat-Blanc. Splittable metamorphic carrier robots. In *14th International Conference on the Synthesis and Simulation of Living Systems*, pages pp. 1–8, New York, US, 2014. The MIT Press.

[BG17] Pau Panareda Busto and Juergen Gall. Open Set Domain Adaptation. In *2017 IEEE International Conference on Computer Vi-*

sion (ICCV), pages 754–763, Venice, October 2017. IEEE.

[BG18] Iago Bonnici and Abdelkader Gouaïch. Formalisation of metamorph Reinforcement Learning. Technical report, LIRMM, Montpellier, France, November 2018.

[Che07] Jenova Chen. Flow in games (and everything else). *ACM Communications*, 50(4):31, April 2007.

[CvG+14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, Yoshua Bengio, and Dzmitry Bahdanau. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, abs/1406.1078, 2014.

[DBME15] Stephane Doncieux, Nicolas Bredeche, Jean-Baptiste Mouret, and Agoston E. (Gusz) Eiben. Evolutionary Robotics: What, Why, and Where to. *Frontiers in Robotics and AI*, 2, March 2015.

[DGD+16] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning Modular Neural Network Policies for Multi-Task and Multi-Robot Transfer. *CoRR*, abs/1609.07088, 2016.

[DXT12] Lixin Duan, Dong Xu, and Ivor W. Tsang. Learning with Augmented Features for Heterogeneous Domain Adaptation. *CoRR*, abs/1206.4660, 2012.

[Elm90] J Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, June 1990.

[FHC+17] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta Learning Shared Hierarchies. *CoRR*, abs/1710.09767, 2017.

[Fra14] Yannick Francillette. *Modèle Adaptatif d'activités Pour Les Jeux Ubiquitaires*. PhD Thesis, Montpellier 2, LIRMM, 2014.

[HA15] Heng Wang and Zubin Abraham. Concept drift detection for streaming data. In *International Joint Conference on Neural Networks*, pages 1–9, Killarney, Ireland, July 2015. IEEE.

[HGC+15] Nadia Hocine, Abdelkader Gouaïch, Stefano A. Cerri, Denis Mottet, Jérome Froger, and Isabelle Laffont. Adaptation in serious games for upper-limb rehabilitation: An approach to improve training outcomes. *User Modeling and User-Adapted Interaction*, 25(1):65–98, March 2015.

[HHCB10] Christopher J. Hanna, Raymond J. Hickey, Darryl K. Charles, and Michaela M. Black. Modular Reinforcement Learning architectures for artificially intelligent agents in complex game environments. In *Computational Intelligence and Games*, pages 380–387, Copenhagen, Denmark, August 2010. IEEE.

[HM00] Robertson Holt and J Mitterer. Examining video game immersion as a flow state. *108th Annual Psychological Association, Washington, DC*, 2000.

[HM10] Maayan Harel and Shie Mannor. Learning from Multiple Outlooks. *CoRR*, abs/1005.0027, 2010.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

[Laz12] Alessandro Lazaric. Transfer in Reinforcement Learning: A Framework and a Survey. In Marco Wiering and Martijn van Otterlo, editors, *Reinforcement Learning*, volume 12, pages 143–173. Springer, 2012.

[LHABL04] Ronan Le Hy, Anthony Arrigoni, Pierre Bessiere, and Olivier Lebeltel. Teaching Bayesian Behaviours to Video Game Characters. *Robotics and Autonomous Systems*, 47:177–185, 2004.

[LHW18] Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, January 2018.

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

[PMJ+16] Mihai Polceanu, Antonio Mora, Jose Jimenez, Cedric Buche, and Antonio Fernandez-Leiva. The Believability Gene in Virtual Bots. In *29th International Flairs*, page 4, Key Largo, Florida, 2016. AAAI Press.

[R C18] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.

[Rin94] Mark Bishop Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, Austin, TX, USA, 1994.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive computation and machine learning series. MIT Press, Cambridge, Mass, 2nd edition, 2018.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015.

[TB10] Fabien Tencé and Cédric Buche. Automatable Evaluation Method Oriented toward Behaviour Believability for Video Games. *CoRR*, abs/1009.0501, 2010.

[TBC+17] Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust Multitask Reinforcement Learning. *CoRR*, abs/1707.04175, 2017.

[TS09] Matthew E. Taylor and Peter Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research*, 10(7):1633–1685, 2009.

[Tsy04] Alexey Tsymbal. The Problem of Concept Drift: Definitions and Related Work. 2004.

[TY03] F. Tanaka and M. Yamamura. Multitask reinforcement learning on the distribution of MDPs. In *International Symposium on Computational Intelligence in Robotics and Automation. Computational Intelligence in Robotics and Automation for the New Millennium*, volume 3, pages 1108–1113, Kobe, Japan, 2003. IEEE.

[XZ18] Ju Xu and Zhanxing Zhu. Reinforced Continual Learning. *CoRR*, abs/1805.12369, 2018.

Learning Methods for RSSI-based Geolocation

Kevin Elgui*[1] et Pascal Bianchi †[1]

[1]Télécom ParisTech

21 juin 2019

Résumé

In this paper, we investigate machine learning approaches addressing the problem of geolocation. First, we review some classical learning methods to build a radio map. In particular, these methods are splitted in two categories, which we refer to as likelihood-based methods and fingerprinting methods. Then, we provide a novel geolocation approach in each of these two categories. The first proposed technique relies on a semi-parametric Nadaraya-Watson estimator of the likelihood, followed by a maximum a posteriori (MAP) estimator of the object's position. The second technique consists in learning a proper metric on the dataset, constructed by means of a Gradient boosting regressor : a k-nearest neighbor algorithm is then used to estimate the position. Finally, all the proposed methods are compared on a data set originated from Sigfox network. The experiments show the interest of the proposed methods, both in terms of location estimation performance, and of ability to build radio maps.

Keywords : LPWA Network, localization, maximum likelihood, metric learning

1 Introduction

Approaches based on the measurement of the received signal strength indicator (RSSI) to geolocate connected objects have witnessed tremendous success since Internet of Things (IoT) is on the rise. In the last few years, IoT has raised a great deal of attention in very diverse fields such as agriculture or health care. Experts agree (in [HL16]) that 30 billion objects will be part of the IoT by 2023 and 40% of these objects will need to be geolocated (e.g for freight transport). To guarantee reliable connectivity between a multitude of connected devices, researchers have been developing various Low Power Wide Area Network (LPWAN) standards [RKS17]. The IoT requires LPWAN standards to support long-range communications. Moreover, ultra-low power consumption is a crucial aspect for the lifetime devices.

Several standard methods such as channel-fingerprinting provide satisfying results in the situation where the propagation channel exhibits enough frequency diversity as shown in [SCGL05]. Nevertheless, in many network of interest, every message transmitted occupies an Ultra Narrow Band (UNB). For instance, for the Sigfox network, the message occupies a band of 100 Hz within the Industrial, Scientific and Medical Band which corresponds to the frequency between 868 MHz and 868.2 MHz in Europe. As a consequence, the geolocation by means of channel-fingerprinting becomes irrelevant because of the absence of frequency diversity.

When the Base Stations (BS's) of a network are time-synchronized, time based approaches as Time Difference Of Arrival (TDOA) provide accurate methods for geolocation [DKD, Fon02]. However, when the BS's are *not* time-synchronized, the collection of the Received Signal Strength Indicator (RSSI) observed at all the BS is, by default, the main source of information allowing to geolocate the source.

In the present paper, we focus on a baseline probabilistic RSSI-only localization algorithm as studied in [YDCPC17, Pat05, JWRS15]. The main challenge comes from the large range of fluctuations of the observed RSSI values, for a given source location. In such data, the observed signals can be very noisy, especially in urban environment (RSSI based methods are often assisted with accelerometers, gyroscopes or Bluetooth beacons to improve their accuracy [YDCPC17]). It may also happen that, due to range limitation or network sensitivity, some messages are not detected by some

*kevin.elgui@gmail.com
†pascal.bianchi@telecom-paristech.fr

BS's. Whereas very few models in the literature chose to regard the information given by the non reception, experience has shown that the performance is increased when the information of non reception is taken into account.

Contributions.

— We provide a method exploiting the advantages of both ensemble methods and k-Nearest Neigbhors ($k-$NN) regressor. The idea, borrowed from [BHS13], is to learn the metric used by the k-NN explicitly for the location estimation task. That is, build a metric to compare two RSSI's vectors, such that the k-NN regressor can chose the most appropriate neighbours for the location estimation task. The metric d is learned such that for a couple of RSSI's vector $(\mathbf{r}, \mathbf{r}')$: $\mathsf{d}(\mathbf{r}, \mathbf{r}')$ is a good predictor of the euclidean distance between the two emitters locations $\|z - z'\|_2$. We propose to learn d as a sum of T regression trees. Those trees are obtained through a XGBoost algorithm. In the end, the k-NN regressor based on the metric d will be used to predict the location. The benefits w.r.t. a classic k-NN regressor are thus twofold :
  — it takes into account the information of reception/non reception of the signal at a BS by replacing the lowest RSSI among all observed RSSI's, as *e.g* in [Pat05] ;
  — it improves the model by optimizing the metric explicitly for the task of geolocation. This drives to better performances of the model (see Section 5).

— We propose a semi-parametric model relying on a relevant likelihood of the RSSI's given the object's position. The shape of the likelihood, is based on a model assumption : given the emitter position, the coordinates of the RSSI vector are independent. The main benefit of this assumption is to allow a low complexity of the model and to make it numerically tractable. The distribution of a RSSI at a given BS, given the location of the emitter will be modeled by a Gaussian distribution. The mean, and the standard deviation of this distribution are obtained by a non-parametric estimator of Nadaraya-Watson type. Finally, the location estimation will be obtained using a Maximum-A-Posteriori (MAP). This proposed estimator enables us to take into account the Boolean variable modeling the reception/non reception of the signal at BS's. The advantages of the provided method are manyfold :
  — it provides good results, even on small training data sets. Moreover, its performances are relatively stable when the number of training points decreases ;
  — it offers a statistical framework through which density level sets and confident regions on the location estimate can be easily computed, when classical machine learning methods (as k-NN), are not able to do so.

— We provide detailed experiments results to compare these methods using real data originated from Sigfox network.

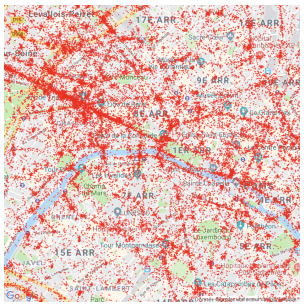


FIGURE 1 – Image of a sample of the locations emitters.

The rest of the paper is organized as follows. In Section 2, we introduce the problem setting. Section 3 investigates several popular geolocation techniques of the literature. Section 4 introduces the proposed predictors. Finally, Section 5 is devoted to the numerical experiments and discussions.

2 Problem setting

The network under consideration is dedicated to long-range and low-power consumption IoT communications. The range of transmission is up to 100 km, and the battery life-time is about 20 years. The network is composed of K fixed BS, say $(\mathrm{BS}_1, \ldots, \mathrm{BS}_K)$, whose respective coordinates $(z_1, \ldots, z_K)$ in the complex plane are known.

Consider a connected device whose position Z is a random variable in some given subset $\mathcal{Z}$, typically an open subset of $\mathbb{R}^2$. The device sends packets/messages which are collected by the neighboring BS. For a given message, each BS k ($k = 1, \ldots, K$) computes a RSSI R_k as the temporal mean of the received signal strength. The RSSI R_k is typically real-valued in a certain subset $\mathcal{R} \subset \mathbb{R}$. However, due to range limitation and network sensitivity, some messages may not be detected by some BS, in which case we just set $R_k = \mathtt{NaN}$, where $\mathtt{NaN}$ stands for an unobserved value. We thus assume that

BS 1	BS 2	...	BS K	Lat	Long
−102	NaN	...	−83	49.15434	2.24928
NaN	−98	...	NaN	48.865584	2.44567

TABLE 1 – Sample from the Sigfox dataset

for every $k = 1, \ldots, K$, R_k is a random variable in the set $\tilde{\mathcal{R}} := \mathcal{R} \bigcup \{\texttt{NaN}\}$.

The aim of this paper is to predict the unknown position Z from the observation of the RSSI-vector

$$\boldsymbol{R} := (R_1, \ldots, R_K)\,.$$

A predictor is a function $\hat{Z} : \tilde{\mathcal{R}}^K \to \mathcal{Z}$. We evaluate the performance of a predictor w.r.t. to the risk $\mathbb{E}(\ell(Z, \hat{Z}(\boldsymbol{R})))$ where $\mathbb{E}$ is the expectation, $\ell : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ is a loss. In typical settings, $\ell(z, \hat{z}) = \|z - \hat{z}\|^2$.

To achieve this task, we assume that the network operator has collected a dataset of fully supervised examples. The dataset is built by gathering observed RSSI's of devices equipped with GPS (see Fig. 1). As represented in Table 1, every row of the dataset corresponds to a message. The features are the RSSI's at the receiving BS's and the label is the GPS coordinates of the transmitting device at the instant when the packet is sent. Formally, the dataset is represented by a collection of n random samples $\mathcal{X}_n := \{(Z^i, \boldsymbol{R}^i) : i = 1, \ldots, n\}$, assumed to be iid copies of $(Z, \boldsymbol{R})$.

3 Review of Geolocation Approaches

In this section, we discuss different off-the-shelf predictors which can be used to solve the geolocation task introduced above.

3.1 Likelihood-based methods

We refer to as Likelihood-based methods the methods which learn from the dataset a likelihood model $p(\boldsymbol{r}|z)$ for the conditional probability of the RSSI vector $\boldsymbol{R}$ given the position Z.

One first learns from the observed data $\mathcal{X}_n$ a mapping $p(\boldsymbol{r}|z)$ which represents the conditional probability density function (pdf) of $\boldsymbol{R}|Z$ namely, the likelihood. To this end, a way is to introduce a parametric likelihood model, such as the path-loss model discussed at the end of this paragraph, and to learn the parameters of this model from the dataset. Non-parametric methods can be used as well (see Section 4). One of the main advantages is that some prior hypotheses on the form of the likelihood $p(\boldsymbol{r}|z)$ can be easily introduced, based on physical considerations. One such hypothesis is the following : The components $R_1, \ldots, R_K$ of the random vector $\boldsymbol{R}$ are independent conditionally to Z. Assumption 3.1 is often used in the literature [KK04], [MBL+09], [Li06]. Discussing its validity is out of the scope of this paper, but we refer the interested reader to [GBSS05] where a independence kernel based test is proposed. Under this hypothesis, the likelihood admits the following decomposition :

$$p(\boldsymbol{r}|z) = \prod_{k=1}^{K} p_k(r_k|z)\,,$$

where $\boldsymbol{r} = (r_1, \ldots, r_K)$ and where $p_1, \ldots, p_K$ are conditional marginals to be learned.

Once the likelihood model $p(\boldsymbol{r}|z)$ has been obtained, the predictor $\hat{Z}(\boldsymbol{R})$ can be easily defined from standard statistical methods. Assume now that a new message arises from the unknown position Z with a RSSI vector $\boldsymbol{R}$. A legitimate (but often computationally intractable) choice is to define the predictor $\hat{Z}(\boldsymbol{R})$ as a minimizer w.r.t. $\hat{z}$ of the estimated risk :

$$\int \ell(z, \hat{z}) p(z|\boldsymbol{R}) dz \tag{1}$$

where, according to the Bayes formula, $p(z|\boldsymbol{R}) \propto p(\boldsymbol{R}|z) p_Z(z)$ and where $p_Z(z)$ is the prior distribution of the r.v. Z supposed to be known (typically uniform on $\mathcal{Z}$ as in [IY10], or inferred from the dataset $\mathcal{X}_n$). As the computation and the minimization of (1) can be difficult, an alternative is to consider the Maximum-a-Posteriori (MAP) estimator given by :

$$\begin{aligned}\hat{Z}_{MAP}(\boldsymbol{R}) &:= \arg\max_{z \in \mathcal{Z}} p(z|\boldsymbol{R}) \\ &= \arg\max_{z \in \mathcal{Z}} \sum_{k=1}^{K} \log p_k(R_k|z) + \log p_Z(z).\end{aligned} \tag{2}$$

To conclude this paragraph, we briefly discuss the broadly used *log-loss* (or *path-loss*) parametric model [YDCPC17], [BOGVB10]. The model is widely used to model the coupling between the received power at the receiver antenna and the distance between the received and emitter. The conditional distribution $p_k(r|z)$ of $R_k|Z$ is supposed to have the form $p_{\theta_k}(r|z)$ where $\theta_k = (P_{0,k}, \nu_k, \sigma_k^2)$ is a triplet of parameters $p_{\theta_k}(\,.\,|z)$ is a Gaussian distribution of variance σ_k^2 and mean $P_{0,k} - 10\nu_k \log_{10} d_v(z, z_k)/d_0$. Here, d_0 is some reference distance and d_v stands for the Vincenty distance, the

parameters $P_{0,k}, \nu_k$ respectively represent the power in dBm at distance d_0 and ν_k is the so-called path-loss exponent. The parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)$ is estimated from the dataset $\mathcal{X}_n$ using a standard maximum likelihood approach.

3.2 Fingerprinting Methods

Fingerprinting methods directly map the vector $\boldsymbol{R}$ into a position Z, typically by means of a supervised learning algorithm. In the following, we present several popular learning algorithms to perform the task of geolocation.

3.2.1 k-Nearest Neighbors (k-NN)

The method is used in [Pat05] in the context of outdoor geolocation. We endow the space of RSSI's vectors with the Euclidean distance. For this purpose, [Pat05] suggests to replace all the `NaN` values either by the lowest RSSI amongst all observed RSSI, or by an arbitrary value (the value -200 is used in [Pat05]). For every K-dimensional RSSI vector $\boldsymbol{R}$, we let $(\boldsymbol{R}^{(1)}, Z^{(1)}), \ldots, (\boldsymbol{R}^{(n)}, Z^{(n)})$ be a reordering of the dataset $\mathcal{X}_n$ such that $\|\boldsymbol{R} - \boldsymbol{R}^{(1)}\| \leq \cdots \leq \|\boldsymbol{R} - \boldsymbol{R}^{(n)}\|$. The unknown position Z is finally estimated by $\hat{Z}(\boldsymbol{R}) := k^{-1} \sum_{i=1}^{k} Z^{(i)}$, where the integer k is an hyperparameter (see [HPS08] for a discussion on the choice of k).

3.2.2 Ensemble Trees Methods

A Random Forest model has been applied as a classifier for a indoor-context geolocation in (see [JWRS15]). In this paper, this method gets better accuracy than a k-NN based method. The goal of such ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve the robustness and the ability to generalize over a single estimator. Two important families are bagging methods such as random forests [Bre01], and boosting methods such as Gradient Tree boosting. The final estimate has the form $\hat{Z}(\boldsymbol{R}) = \sum_{t=1}^{T} f_t(\boldsymbol{R})$ where T is an integer and $f_1, \ldots, f_T$ are regression trees learned on the dataset $\mathcal{X}_n$ by one of the above methods.

4 Proposed Geolocation Methods

4.1 Semi-Parametric Likelihood-Based Method

We propose the following semi-parametric likelihood model for the conditional density $p(\boldsymbol{r}|z)$,of $\boldsymbol{R}$ given Z. As often in geolocation [KK04, MBL+09, Li06, YDCPC17], we strongly rely on the conditional independence Assumption 3.1. Using the later hypothesis, it is sufficient to provide a model for the marginal conditional distributions $p_k(r_k|z)$ of R_k given Z, for every $k = 1, \ldots, K$. Here, we recall that R_k is a random variable over the set $\mathbb{R} \cup \{\texttt{NaN}\}$. Densities are thus considered w.r.t. the reference measure $\lambda + \delta_{\texttt{NaN}}$ where λ is the Lebesgue measure and $\delta_{\texttt{NaN}}$ is the Dirac measure at the `NaN`-value. We define $\pi_k : \mathcal{Z} \to [0, 1]$ as

$$\pi_k(z) := \mathbb{P}(R_k = \texttt{NaN}|Z = z)$$

and we constrain the model by assuming that, given Z and given that $R_k \neq \texttt{NaN}$, R_k follows a Gaussian distribution whose mean and variance are respectively denoted by $m_k(z)$ and $\sigma_k^2(z)$:

$$\begin{aligned} m_k(z) &:= \mathbb{E}(R_k|Z = z, R_k \neq \texttt{NaN}) \\ \sigma_k^2(z) &:= \mathrm{Var}(R_k|Z = z, R_k \neq \texttt{NaN}) \,. \end{aligned}$$

We denote by $r \mapsto \Phi(r; m, \sigma^2)$ the normal density of mean m and variance σ^2. We summarize our model is as follows :

1. $R_1, \ldots, R_K$ are independent given Z ;
2. For every k,

$$\begin{aligned} \mathbb{P}(R_k \in dr|Z) = \pi_k(Z)\delta_{\texttt{NaN}}(dr) \\ + (1 - \pi_k(Z))\Phi(r; m_k(Z), \sigma_k^2(Z))dr \,. \end{aligned}$$

Based on this model, the likelihood $p(\boldsymbol{r}|z)$ is fully determined by the mappings π_k, m_k and σ_k^2 for all $k = 1, \ldots, K$. The remaining task is to estimates these quantities using our dataset $\mathcal{X}_n$. To this end, we propose to use a non-parametric approach, and to replace these mappings with their Nadaraya-Watson estimates [Tsy13]. Let $K : \mathcal{Z} \to \mathbb{R}_+$ be a kernel, i.e., non-negative, symmetric function integrating to one, and let $h > 0$ be a scalar (the so-called *bandwidth*). Define $K_h(z) = h^{-1}K(h^{-1}z)$ for all $z \in \mathcal{Z}$. The Nadaraya-

Watson estimates are respectively given for every k by

$$\hat{\pi}_k(z) := n^{-1} \sum_{i=1}^{n} \mathbb{1}_{\texttt{NaN}}(R_k^i) K_h\left(Z^i - z\right)$$

$$\hat{m}_k(z) := D_k(z)^{-1} \sum_{i=1}^{n} \mathbb{1}_{\mathbb{R}}(R_k^i) R_k^i K_h\left(Z^i - z\right)$$

$$\hat{\sigma}_k^2(z) := D_k(z)^{-1} \sum_{i=1}^{n} \mathbb{1}_{\mathbb{R}}(R_k^i)(R_k^i - m_k(z))^2 K_h\left(Z^i - z\right)$$

where $D_k(z) := \sum_{i=1}^{n} \mathbb{1}_{\mathbb{R}}(R_k^i) K_h\left(Z^i - z\right)$. Under standard technical conditions, $\hat{\pi}_k$, $\hat{m}_k$ and $\hat{\sigma}_k^2$ converge uniformly towards π_k, m_k and σ_k^2 as $n \to \infty$ and $nh \to \infty$ [Tsy13]. Finally, the MAP location estimator can be written as :

$$\hat{Z}(\boldsymbol{R}) = \arg\max_{z \in \mathcal{Z}} \sum_{k \in \mathcal{I}_{\boldsymbol{R}}} (1 - \hat{\pi}_k(z)) \log \Phi(R_k; \hat{m}_k(z), \hat{\sigma}_k^2(z)) + \sum_{k \in \mathcal{I}_{\boldsymbol{R}}^c} (1 - \hat{\pi}_k(z)) + \log \hat{p}_Z(z)\,,$$

where $\mathcal{I}_{\boldsymbol{R}} := \{k = 1, \ldots, K : R_k \neq \texttt{NaN}\}$ stands for the set of the receiving BS's, and where $\hat{p}_Z(z)$ stands for an estimation of the prior on Z, which we suggest to estimate from the dataset $\mathcal{X}_n$ through the kernel density estimator :

$$\hat{p}_Z(z) = n^{-1} \sum_{i=1}^{n} K_h\left(Z^i - z\right)\,.$$

4.2 Metric-Learning Fingerprinting Method

In this paragraph, we tackle the problem of learning an adapted metric (see [XWNW16]) on $\mathcal{R}^K$ to improve basic k-NN using the standard Euclidean distance. We recall that NaN values are here replaced by a fixed real value as discussed in Section 3.2. The idea is to build a mapping $\mathsf{d} : \mathcal{R}^K \times \mathcal{R}^K \to [0, +\infty)$ such that *close RSSI* (w.r.t. to the metric d) *correspond to close object positions* (w.r.t. to the Vincenty distance d_v on $\mathcal{Z}$). In that sense, a "good" metric (see Fig. 2) is a mapping d for which the empirical risk

$$ER_n(\mathsf{d}) := \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\mathsf{d}(R^i, R^j) - d_v(Z^i, Z^j)\right)^2$$

is small. The main trick, borrowed from [XJXC12], [BHS13] is to search for a mapping d minimizing $\mathbb{E}R_n(\mathsf{d})$ within a relevant hypothesis class. More precisely, we search for d under the form

$$\mathsf{d}(\boldsymbol{r}, \boldsymbol{r}') := \sum_{t=1}^{T} f_t(\varphi(\boldsymbol{r}, \boldsymbol{r}'))\,,$$

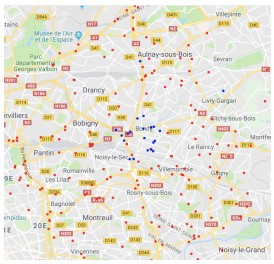


FIGURE 2 – Scatter plots of the k nearest neighbors. In red, a sample of the 200 neighbors according to the euclidean distance. In blue, the 25 neighbors according to the learned metric. In green stars, the true position of the emitter.

where $f_1, \ldots, f_T$ is a collection of T regression trees, and where $\varphi : \mathcal{R}^K \times \mathcal{R}^K \to \mathbb{R}^K \times \mathbb{R}^K$ is given by :

$$\varphi(\boldsymbol{r}, \boldsymbol{r}') := \begin{pmatrix} |\boldsymbol{r} - \boldsymbol{r}'| \\ \frac{1}{2}(\boldsymbol{r} + \boldsymbol{r}') \end{pmatrix}.$$

In practice, the minimization of $ER_n(\mathsf{d})$ w.r.t. $f_1, \ldots, f_T$ is untractable. An alternative is to use a Random Forest or an XGboost regressor, which separately optimizes the T regression trees. In practice, the learning stage is thuse as follows :

- Compute the pairwise features $\varphi(\boldsymbol{R}^i, \boldsymbol{R}^j)$ for all couples (i, j) in the dataset ;
- Use a regression tree ensemble method to predict the labels $d_v(Z^i, Z^j)$ based on the features $\varphi(\boldsymbol{R}^i, \boldsymbol{R}^j)$.

Note that the obtained mapping d, though symmetric, is not mathematically speaking a metric. This point is however irrelevant regarding the application of interest. Given the obtained metric and given an observed RSSI vector $\boldsymbol{R}$, the k-NN estimate of Z is computed as in Section 3.2.

5 Numerical experiments

5.1 Performance Analysis

To compare the performances of the different methods, we use the Sigfox dataset $\mathcal{X}_n$ composed of $n = 1.5 \cdot 10^6$ observations. The dataset was randomly split in a training subset (90%) and a test subset (10%). The training subset was used to perform cross-validation (each fold containing 10% of the training set) in order to find the optimal parameters of our algorithms. The test subset was employed to evaluate the accuracy of the methods in competition.

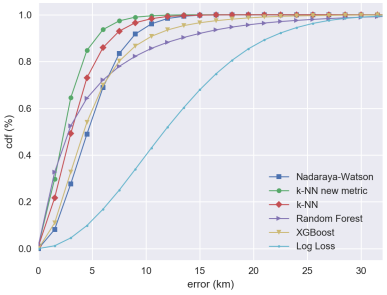


FIGURE 3 – Comparisons of the presented methods in terms of the c.d.f. of errors.

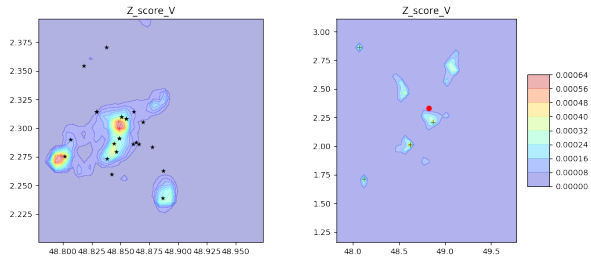


FIGURE 4 – Heat map of the position $Z|\boldsymbol{R}$ for two different observations of $\boldsymbol{R}$. The red dots show the true positions. The black dots are the observed positions in the test dataset corresponding for the same observations of $\boldsymbol{R}$.

To compute the errors we employ the Vincenty distance between estimated and actual location. Fig. 3 shows the cumulative distribution function of the estimation error for all the presented methods. The k-NN with the learned metric turns out to outperform the other methods of the paper. By contrast, the Log Loss model is not relevent for this noisy urban dataset.

5.2 Heat Map estimation

A major benefit of the Semi-Parametric Likelihood-Based method is that density level sets can be computed easily. Thanks to the statistical framework, this method is able to evaluate the probability density of $Z|\boldsymbol{R}$ at all $z \in \mathcal{Z}$. This density level sets are regions in which Z is most likely to lie given the observation of $\boldsymbol{R}$. This is shown in Fig. 4 where further information is provided on the uncertainty of the estimation.

Conclusion

In this paper, we investigated machine learning approaches addressing the problem of geolocation. We presented most popular methods that can be found in the literature. Then, we proposed two new techniques : one based on a likelihood and the other on a learned metric for a k-NN. To compare these methods, 1,5M observations were collected from the Sigfox network. Results have shown that the metric learning method has obtained the highest accuracy on this dataset. As for the semi-parametric method, it goes beyond the simple estimation by providing heat maps and level sets, making it, a suitable methods for industrial applications.

Références

[BHS13] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv :1306.6709*, 2013.

[BOGVB10] Mussa Bshara, Umut Orguner, Fredrik Gustafsson, and Leo Van Biesen. Fingerprinting localization in wireless networks based on received-signal-strength measurements : A case study on wimax networks. *IEEE Transactions on Vehicular Technology*, 59(1) :283–294, 2010.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

[DKD] B Denis, J Keignart, and N Daniele. Impact of nlos propagation upon ranging precision in uwb systems. Citeseer.

[Fon02] Robert J Fontana. Experimental results from an ultra wideband precision geolocation system. In *Ultra-Wideband, Short-Pulse Electromagnetics 5*, pages 215–223. Springer, 2002.

[GBSS05] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

[HL16] Chin-Lung Hsu and Judy Chuan-Chuan Lin. An empirical examination of consumer adoption of internet of things services : Network externalities and concern

for information privacy perspectives. *Computers in Human Behavior*, 62 :516–527, 2016.

[HPS08] Peter Hall, Byeong U Park, and Richard J Samworth. Choice of neighbor order in nearest-neighbor classification. *The Annals of Statistics*, pages 2135–2152, 2008.

[IY10] Mohamed Ibrahim and Moustafa Youssef. Cellsense : A probabilistic rssi-based gsm positioning system. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5. IEEE, 2010.

[JWRS15] Esrafil Jedari, Zheng Wu, Rashid Rashidzadeh, and Mehrdad Saif. Wi-fi based indoor location positioning employing random forest classifier. In *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*, pages 1–5. IEEE, 2015.

[KK04] Kamol Kaemarungsi and Prashant Krishnamurthy. Properties of indoor received signal strength for wlan location fingerprinting. In *Mobile and Ubiquitous Systems : Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 14–23. IEEE, 2004.

[Li06] Xinrong Li. Rss-based location estimation with unknown pathloss model. *IEEE Transactions on Wireless Communications*, 5(12), 2006.

[MBL+09] Santiago Mazuelas, Alfonso Bahillo, Ruben M Lorenzo, Patricia Fernandez, Francisco A Lago, Eduardo Garcia, Juan Blas, and Evaristo J Abril. Robust indoor positioning provided by real-time rssi values in unmodified wlan networks. *IEEE Journal of selected topics in signal processing*, 3(5) :821–831, 2009.

[Pat05] Neal Patwari. *Location estimation in sensor networks*. PhD thesis, University of Michigan, 2005.

[RKS17] Usman Raza, Parag Kulkarni, and Mahesh Sooriyabandara. Low power wide area networks : An overview. *IEEE Communications Surveys & Tutorials*, 19(2) :855–873, 2017.

[SCGL05] Guolin Sun, Jie Chen, Wei Guo, and KJ Ray Liu. Signal processing techniques in network-aided positioning : a survey of state-of-the-art positioning designs. *IEEE Signal Processing Magazine*, 22(4) :12–23, 2005.

[Tsy13] Alexandre Tsybakov. Apprentissage statistique et estimation non-paramétrique. Course, 2013.

[XJXC12] Caiming Xiong, David Johnson, Ran Xu, and Jason J Corso. Random forests for metric learning with implicit pairwise position dependence. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 958–966. ACM, 2012.

[XWNW16] Yaqin Xie, Yan Wang, Arumugam Nallanathan, and Lina Wang. An improved k-nearest-neighbor indoor localization method based on spearman distance. *IEEE Signal Process. Lett.*, 23(3) :351–355, 2016.

[YDCPC17] Simon Yiu, Marzieh Dashti, Holger Claussen, and Fernando Perez-Cruz. Wireless rssi fingerprinting localization. *Signal Processing*, 131 :235–244, 2017.

From the *Token* to the *Review*: A Hierarchical Multimodal approach to Opinion Mining

Alexandre Garcia*[1], Pierre Colombo*[2], Florence d'Alché-Buc[3], Slim Essid[4], et Chloé Clavel[5]

[1,2,3,4,5]Télécom ParisTech

30 mai 2019

Résumé

The task of predicting fine grained user opinion based on spontaneous spoken language is a key problem arising in the development of Computational Agents as well as in the development of social network based opinion miners. Unfortunately, gathering reliable data on which a model can be trained is notoriously difficult and existing works rely only on coarsely labeled opinions. In this work we aim at bridging the gap separating fine grained opinion models already developed for written language and coarse grained models developed for spontaneous multimodal opinion mining. We take advantage of the implicit hierarchical structure of opinions to build a joint fine and coarse grained opinion model that exploits different views of the opinion expression. The resulting model shares some properties with attention-based models and is shown to provide competitive results on a recently released multimodal fine grained annotated corpus.

1 Introduction

Recent years have witnessed the increasing popularity of social networks and video streaming platforms. People heavily rely on these channels to express their opinions through video-based discussions or reviews. Whereas such opinionated data has been widely studied in the context of written customer reviews [Liu12] crawled on websites such as Amazon [HL04] and IMDB [MDP+11], only a few studies have been proposed in the case of video-based reviews. Such multimodal data has been shown to provide a mean to disambiguate some hard to understand opinion expressions such as irony and sarcasm [AEHP03] and contains crucial information indicating the level of engagement and the persuasiveness of the speaker [CC16, BCE19, NGK+16]. A key problem in this context is the lack of availability of fine grained opinion annotation *i.e.* annotations performed at the token or short span level and highlighting on the components of the structure of opinions. Indeed whereas such resources have been gathered in the case of textual data and can be used to deeply understand the expression of opinions [WWC05, PGP+16], the different attempts at annotating multimodal reviews have shown that reaching good annotator agreement is nearly impossible at a fine grained level. This results from the disfluent aspect of spontaneous spoken language making it difficult to choose opinions' annotation boundaries [GEDBC19, LC15b]. Thus the price to pay to gather reliable data is the definition of an annotation scheme focusing on coarse grained information such as long segment categorization as done by [ZZPM16a] or review level annotation [PSC+14]. Building models able to predict fine grained opinion information in a multimodal setting is in fact of high importance in the context of designing human–robot interfaces [LC16]. Indeed the knowledge of opinions decomposed over a set of polarities associated to some targets is a building block of automatic human understanding pipelines [LC15a]. The following work is motivated by the following observations :

— Despite the lack of reliability of fine grained labels collected for multimodal data, the redundancy of the opinion information contained at different granularities can be leveraged to reduce the inherent noise of the labelling process and to build improved opinion predictors. We build a model that takes advantage of this property and joinlty models the different components of an opinion.

— Hierarchical multi-task language models have been recently shown to improve upon the single tasks' models [SWR18]. A careful choice of the tasks and the order in which they are sequentially presented to the model has been proved to be the key to build competitive predictors. It is not clear whether such type of hierarchical model could be adapted to handle multi-

*equal contribution

modal data with the state of the art neural architectures [ZLP+18, ZLM+18]. We discuss in the experimental section the strategies and models that are adapted to the multimodal opinion mining context.

— In the case where no fine grained supervision is available, the attention mechanism [VSP+17] provides a compelling alternative to build models generating interpretable decisions with token-level explanations [HFV+18]. In practice such models are notoriously hard to train and require the availability of very large datasets. On the other hand, the injection of fine-grained polarity information has been shown to be a key ingredient to build competitive sentiment predictors by [SPW+13]. Our hierarchical approach can be interpreted under the lens of attention-based learning where some supervision is provided at training to counterbalance the difficulty of learning meaningful patterns with spoken language data. We specifically experimentally show that providing this supervision is here necessary to build competitive predictors due to the limited number of data and the difficulty to extract meaningful patterns from it.

2 Background on fine grained opinion mining

The computational models of opinion are grounded in a linguistic framework defining how these objects can be structured over a set of interdependent functional parts. In this work we focus on the model of [MW13] that defines the expression of opinions as an *evaluation* towards an object. The expression of such evaluations can be summarized by the combination of three components : a *source* (mainly the speaker) expressing a statement on a *target* identifying the entity evaluated and a *polarized expression* making the attitude of the source explicit. In the literature, the task of finding the words indicating these components and categorizing them using a set of predefined possible targets and polarities has been studied under the name of Aspect Based Sentiment Analysis (ABSA) and popularized by the SEMEVAL campaigns [PGP+16]. They defined a set of tasks including sentence-level prediction. *Aspect Category Detection* consists in finding the target of an opinion from a set of possible entities ; *Opinion Target Expression* is a sequence tagging problem where the goal is to find the word indicating this entity ; and *Sentiment Polarity* recognition is a classification task where the predictor has to determine whether the underlying opinion is positive, negative or neutral. Such problems have also been extended at the text level (*text-level ABSA*) where the participants were asked to predict a set of tuples (Entity category, Polarity level) summarizing the opinions contained in a review. In this work we adapt these tasks to a recently released fine-grained multimodal opinion mining corpus and study a category of hierarchical neural architecture able to jointly perform *token*-level, *sentence*-level and review-level predictions. In the next sections, we present the data available and the definition of the different tasks.

3 Data description and model

This work relies on a set of fine and coarse grained opinion annotations gathered for the Persuasive Opinion Multimedia (POM) corpus presented in [GEDBC19]. The dataset is composed of 1000 videos carrying a strong opinion content : in each video, a single speaker in frontal view makes a critique of a movie that he/she has watched. The corpus contains 372 unique speakers and 600 unique movie titles. The opinion of each speaker has been annotated at 3 levels of granularity as shown in Figure 1.

At the finest (*Token*) level, the annotators indicated for each token whether it is responsible for the understanding of the polarity of the sentence and whether it describes the target of an opinion. On top of this, a span-level annotation contains a categorization of both the target and the polarity of the underlying opinion in a set of predefined possible target *entities* and polarity *valences*. At the review level (or *text*-level since the annotations are aligned with the tokens of the transcript), an overall score describes the attitude of the reviewer about the movie.

As [GEDBC19] have shown that the boundaries of span-level annotations are unreliable, we relax the corresponding boundaries at the sentence level. This *sentence* granularity is in our data the intermediate level of annotation between the *token* and the *text*. In practice, these intermediate level labels can be modeled by tuples such as the one provided in the *text-level ABSA* SEMEVAL task which are given for each sentence in the dataset. In what follows, we will refer to the problem of predicting such information as the *sentence level*-prediction problem. Details concerning the determination of the sentence boundaries and the associated pre-processing of the data are given in the supplementary material.

The representation described above can be naturally converted into a mathematical representation : A review $\mathbf{x}^{(i)}$, $i \in \{1, \ldots, N\}$ is made of S_i sentences each containing W_{S_i} words. Thus the canonical feature representation of a review is the following $\mathbf{x}^{(i)} = \{\{x^{(i)}_{1,1}, \ldots, x^{(i)}_{1,W_{S_1}}\}, \ldots, \{x^{(i)}_{S_i,1}, \ldots, x^{(i)}_{S_i,W_{S_i}}\}\}$, where each x is the feature representation of a spoken word corresponding to the concatenation of a textual, audio and video feature representation. It has been shown in [ZLP+18, ZZPM16a, ZZPM16b] that whereas the textual modality carries the most information, taking into account

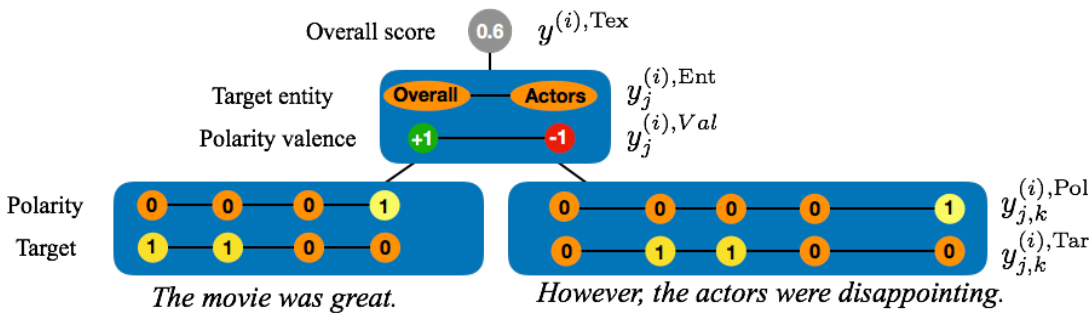


FIGURE 1 – Structure of an annotated opinion

video and audio modalities is mandatory to obtain state of the art results on sentiment analysis problems. Based on this input description, the learning task consists in finding a parameterized function $g_\theta : \mathcal{X} \to \mathcal{Y}$ that predicts various components of an opinion $\mathbf{y} \in \mathcal{Y}$ based on an input review $\mathbf{x} \in \mathcal{X}$. The parameters of such a function are obtained by minimizing an empirical risk :

$$\hat{\theta} = \min_\theta \sum_{i=1}^{N} l(g_\theta(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}), \tag{1}$$

where l is a non-negative loss function penalizing wrong predictions. In general the loss l is chosen as a surrogate of the evaluation metric whose purpose is to measure the similarity between the predictions and the true labels. In the case of complex objects such as opinions, there is no natural metric for measuring such proximity and we rely instead on distances defined on substructures of the opinion model. To introduce these distances, we first decompose the label-structures following the model previously described :

— *Token*-level labels are represented by a sequence of 2-dimensional binary label vectors $y_{j,k}^{(i),\text{Tok}} = \begin{pmatrix} y_{j,k}^{(i),\text{Pol}} \\ y_{j,k}^{(i),\text{Tar}} \end{pmatrix}$ where $y_{j,k}^{(i),\text{Pol}}$ is a binary variable indicating whether the kth word of the sentence j in review i is a word indicating the polarity of an opinion (respectively the target of an opinion for $y_{j,k}^{(i),Tar}$).

— *Sentence*-level labels carry 2 pieces of information : the categorization of the target *entities* mentioned in an opinion expressed is represented by an E dimensional binary vector $y_j^{(i),\text{Ent}}$ where each component encodes the presence of an entity among E possible values. And the polarity of the opinions contained in the sentence are represented by a 4-dimensional one-hot vector $y_j^{(i),\text{Val}}$ encoding the possible *valences* : *Positive*, *Negative*, *Neutral/Mixed* and *None*. Thus the sentence level label $y_j^{(i),\text{Sent}}$ is the concatenation of the two representations presented above : $y_j^{(i),\text{Sent}} = \begin{pmatrix} y_j^{(i),\text{Ent}} \\ y_j^{(i),\text{Val}} \end{pmatrix}$

— *Text*-level labels are composed of a single continuous score obtained for each review $y^{(i),Tex}$ summarizing the overall rating given by the reviewer to the entity described.

Based on these representations, we define a set of losses, $l^{(\text{Tok})}, l^{(\text{Sent})}, l^{(\text{Tex})}$ dedicated to measuring the similarity of each substructure prediction, $\hat{\mathbf{y}}^{(\text{Tok})}, \hat{\mathbf{y}}^{(\text{Sent})}, \hat{\mathbf{y}}^{(\text{Tex})}$ with the ground-truth. In the case of binary variables and in the absence of prior preference between targets and polarities, we use the negative log-likelihood for each variable. Each task loss is then defined as the average of the negative log-likelihood computed on the variables that compose it. For continuous variables, we use the mean squared error as the task loss. Consequently the losses to minimize can be expressed as :

$$\begin{aligned} l^{(Tok)}(\mathbf{y}^{\text{Tok}}, \hat{\mathbf{y}}^{\text{Tok}}) &= -\frac{1}{2}\sum_i ((\mathbf{y}_i^{Pol} \log(\hat{\mathbf{y}}_i^{Pol}) + \\ &\quad \mathbf{y}_i^{Tar} \log(\hat{\mathbf{y}}_i^{Tar})), \\ l^{(Sent)}(\mathbf{y}^{Sent}, \hat{\mathbf{y}}^{Sent}) &= -\frac{1}{2}\sum_i (\mathbf{y}_i^{\text{Ent}} \log(\hat{\mathbf{y}}_i^{\text{Ent}}) + \\ &\quad \mathbf{y}_i^{\text{Val}} \log(\hat{\mathbf{y}}_i^{\text{Val}})), \\ l^{(Tex)}(y^{\text{Tex}}, \hat{y}^{\text{Tex}}) &= (y^{\text{Tex}} - \hat{y}^{\text{Tex}})^2, \end{aligned}$$

Following previous works on multi-task learning [AEP07, Rud17], we argue that optimizing simultaneously the risks derived from these losses should improve the results, compared to the case where they are treated separately, due to the knowledge transferred across tasks. In the multi-task setting, the loss l derived from a set of task losses $l^{(t)}$, is a convex combination of these different task losses. Here the tasks corresponds to each granularity level : $t \in \text{Tasks} = \{Tok, Sent, Tex\}$ weighted according to a set of task weights

3

λ_t :

$$l(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\sum_{t \in \text{Tasks}} \lambda_t l^{(t)}(\mathbf{y}^t, \hat{\mathbf{y}}^t)}{\sum_{t \in \text{Tasks}} \lambda_t}, \ \forall \lambda_t \geq 0. \quad (2)$$

Optimizing this type of objectives in the case of neural hierarchical predictors requires building some strategy in order to train the different parts of the model : the low level parts as well as the abstract ones. We discuss such issue in the next section.

4 Learning strategies for multitask objectives

The main concern when optimizing objectives of the form of Équation 2 comes from the variable difficulty in optimizing the different objectives $l^{(t)}$. Previous works [SWR18] have shown that a careful choice of the order in which they are introduced is a key ingredient to correctly train deep hierarchical models. In the case of hierarchical labels, a natural hierarchy in the prediction complexity is given by the problem. In the task at hand, coarse grained labels are predicted by taking advantage of the information coming from predicting fine grained ones. The model processes the text by recursively merging and selecting the information in order to build an abstract representation of the review. In Experiment 1 we show that incorporating these fine grained labels into the learning process is necessary to obtain competitive results from the resulting predictors. In order to gradually guide the model from easy tasks to harder ones, we parameterize each λ_t as a function of the number of epochs of the form $\lambda_t^{(n_{\text{epoch}})} = \lambda_{\max} \frac{\exp((n_{\text{epoch}} - Ns_t)/\sigma)}{1+\exp((n_{\text{epoch}} - Ns_t)/\sigma)}$ where Ns_t is a parameter devoted to task t controlling the number of epochs after which the weight switches to $\lambda_{\max}$ and σ is a parameter controlling the slope of the transition. We construct 4 strategies relying on smooth transitions from a low state $\lambda_t^{(0)} = 0$ to a high state $\lambda_t^{(Ns_t)} = \lambda_t^{\max}$ of each task weight varying with the number of epochs. The different strategies described below are illustrated in the supplementary material.

— Strategy 1 (S1) consists in optimizing the different objectives one at a time from the easiest to the hardest. It consists in first moving vector $(\lambda_{Token}, \lambda_{Sentence}, \lambda_{Text})^T$ values from $(1,0,0)^T$ to $(0,1,0)^T$ and then finally to $(0,0,1)^T$. The underlying idea is that the low level labels are only useful as an initialization point for higher level ones.

— Strategy 2 (S2) consists in adding sequentially the different objectives to each other from the easiest to the hardest. It goes from a word only loss $(\lambda_{Token}, \lambda_{Sentence}, \lambda_{Text})^T = (\lambda_{Token}^{(N)}, 0, 0)^T$ and then adds the intermediate objectives by setting $\lambda_{Sentence}$ to $\lambda_{Sentence}^{(N)}$ and then λ_{Text} to $\lambda_{Text}^{(N)}$. This strategy relies on the idea that keeping a supervision on low level labels has a regularizing effect on high level ones. Note that this strategy and the two following require a choice of the stationary weight values $\lambda_{Token}^{(N)}, \lambda_{Sentence}^{(N)}, \lambda_{Text}^{(N)}$.

— Strategy 3 (S3) is similar to (S2) except that the *sentence* and *text* weights are simultaneously increased. This strategy and the following one are introduced to test whether the order in which the tasks are introduced has some importance on the final scores.

— Strategy 4 (S4) is also similar to (S2) except that *text*-level supervision is introduced before the *sentence*-level one. This strategy uses the intermediate level labels as a way to regularize the video level model that would have been learned directly after the *token*-level supervision

These strategies can be implemented in any stochastic gradient training procedure of objectives (Équation 2) since it only requires modifying the values of the weight at the end of each epoch. In the next section, we design a neural architecture that jointly predicts opinions at the three different levels, *i.e.* the *token*, *sentence* and *text* levels, and discuss how to optimize multitask objectives built on top of opinion-based output representations.

5 Architecture

Before digging into the model description, we introduce the set of hidden variables $h^{(i),\text{Tex}}, h_j^{(i),Sent}, h_{j,k}^{(i),\text{Tok}}$ corresponding to the unconstrained scores used to predict the outputs : $\hat{y}^{(i),\text{Tex}} = \sigma^{\text{Tex}}(W^{\text{Tex}} h^{(i),\text{Tex}} + b^{\text{Tex}})$, $\hat{y}_j^{(i),\text{Sent}} = \sigma^{\text{Sent}}(W^{\text{Sent}} h_j^{(i)^{\text{Sent}}} + b^{\text{Sent}})$, $\hat{y}_{j,k}^{(i),\text{Tok}} = \sigma^{\text{Tok}}(W^{\text{Tok}} h_{j,k}^{(i),\text{Tok}} + b^{\text{Tok}})$, where the W and b are some parameters learned from data and the σ are some fixed almost everywhere differentiable functions ensuring that the outputs "match" the inputs of the loss function. In the case of binary variables for example, it is chosen as the sigmoid function $\sigma(x) = \exp(x)/(1+\exp(x))$. From a general perspective, a hierarchical opinion predictor is composed of 3 functions $g^{\text{Tex}}, g^{\text{Sent}}, g^{\text{Tok}}$ encoding the dependency across the levels :

$$\begin{aligned} h_{j,k}^{(i),\text{Tok}} &= g_{\theta^{\text{Tok}}}^{\text{Tok}}(x_{j,:}^{(i),\text{Tok}}), \\ h_j^{(i)^{\text{Sent}}} &= g_{\theta^{\text{Sent}}}^{\text{Sent}}(h_{j,:}^{(i)^{\text{Tok}}}), \\ h^{(i)^{\text{Tex}}} &= g_{\theta^{\text{Tex}}}^{\text{Tex}}(h_{:}^{(i)^{\text{Sent}}}). \end{aligned}$$

In this setting, low level hidden representations are shared with higher level ones. A large body of work has focused on the design of the g functions in the case of multimodal inputs. In this work we exploit state of the art sequence encoders to build our hidden representations that we detail below. The

mathematical expression of the models and a more in depth description are provided in the supplementary material.

— Gated Recurrent Units (GRU) [CVMG+14] especially when coupled with a self attention mechanism have been shown to provide state of the art results on tasks implying the encoding or decoding of a sentence in or from a fixed size representation. Such a problem is encountered in automatic machine translation [LPM15], automatic summarization [NZZ17] or image captioning and visual question answering [AHB+18]. We experiment with both models mixing the 3 concatenated input feature modalities (GRU model in Experiment 1) and a model carrying 3 independent GRU with a hidden state per modality (Ind GRU models).

— The Multi-attention Recurrent Network (MARN) proposed in [ZLP+18] extends the traditional Long Short Term Memory (LSTM) sequential model by both storing a view specific dynamic (similar to the LSTM one) and by taking into account cross-view dynamics computed from the signal of the other modalities. In the original paper, this cross-view dynamic is computed using a multi-attention bloc containing a set of weights for each modality used to mix them in a joint hidden representation. Such a network can model complex dynamics but does not embed a mechanism dedicated to encoding very long-range dependencies.

— Memory Fusion Networks (MFN) are a second family of multi-view sequential models built upon a set of LSTM per modality feeding a joint delta memory. This architecture has been designed to carry some information in the memory even with very long sequences due to the choice of a complex retain / forget mechanism.

The 3 models described previously build a hidden representation of the data contained in each sequence. The transfer from one level of the hierarchy to the next coarser one requires building a fixed length representation summarizing the sequence. Note that in the case of the MARN and the MFN, the model directly creates such a representation. We present the strategies that we deployed to pool these representations in the case of the GRU sequential layer.

— Last state representation : Sequential models build their inner state based on observations from the past. One can thus naturally use the hidden state computed at the last observation of a sequence to represent the entire sequence. In our experiments, this is the representation chosen for the GRU and Ind GRU models.

— Attention based sequence summarization : Another technique consists in computing a weighted sum of the hidden states of the sequence. The attention weights can be learned from data to focus on the important parts of the sequence only and avoid building too complex inner representations. An example of such a technique successfully applied to the task of text classification based on 3 levels of representation can be found in [YYD+16]. In our experiments, we implemented the attention model for predicting only the *Sentence*-level labels (model Ind GRU + att Sent) and the *Sentence* and *Text*-level labels by sharing a common representation (Ind GRU + att model).

All the resulting architectures extend the existing hierarchical models by enabling the fusion of multimodal information at different granularity levels while maintaining the ability to introduce some supervision at any level.

6 Experiments

In this section we propose 3 sets of experiments that show the superiority of our model over existing approaches with respect to the difficulties highlighted in the introduction, and explore the question of the best way to train hierarchical models on multimodal opinion data.

All the results presented below have been obtained on the recently released fine grained annotated POM dataset [GEDBC19]. The input features are computed using the CMU-Multimodal SDK : We represented each word by the concatenation of the 3 feature modalities. The textual features are chosen as the 300-dimensional pre-trained Glove embeddings [PSM14]. The acoustic and visual features have been obtained by averaging the descriptors computed following [PSC+14] during the time of pronunciation of each spoken word. These features include MFCC and pitch descriptors for the audio signals. For the video descriptors, posture, head and gaze movement are taken into account. As far as the output representations are concerned, we merely re-scaled the *Text*-level polarity labels in the [0,1] range.

The results are reported in terms of mean average error (MAE) for the continuous labels and micro F1 score $\mu F1$ for binary labels. We used the provided train, val and test set and describe for each experiment the training procedure and displayed values below. More detail concerning the preprocessings and architectures can be found in the supplementary material.

6.1 Experiment 1 : Which architecture provides the best results on the task of fine grained opinion polarity prediction ?

In this first section, we describe our protocol to select an architecture devoted to performing fine grained multimodal opinion prediction. In order to focus our analysis on a restricted set of possible models, we only treat the polarity prediction problem in this section and selected the architectures that provided the best review-level scores (*i.e.* with

lowest mean average prediction error). Taking into account the entity categories would only bring an additional level of complexity that is not necessary in this first model selection phase. Building upon previous works [ZLM+18], we use the MFN model as our *sentence*-level sequential model since it has been shown to provide state of the art results on *text*-level prediction problems on the POM dataset. For the *token*-level model, we test different state of the art models able to take advantage of the multimodal information. Our architecture is built upon the *token*-level encoders presented in section 5 : the MFN, MARN and independent GRUs. Our baseline is computed similarly to [ZLP+18] : we represent each sentence by taking the average of the feature representation of the Tokens composing it. The best results reported were obtained after a random search on the parameters and presented in Table 1. In the top row, we report results obtained when only using the *text*-level labels to train the entire network. The baseline consisting in representing each sentence by the average of its tokens representation strongly outperforms all the other results. This is due to the moderate size of the training set (600 videos) which is not enough to learn meaningful fine grained representations. In the second part, we introduce some supervision at all levels and found that a choice of $\lambda_{Tok} = 0.05,\ \lambda_{Sent} = 0.5,\ \lambda_{Tex} = 1$ being respectively the *token*, *sentence* and *text* weights provides the best *text*-level results. This combination reflects the fact that the main objective (*text*-level) should receive the highest weight but low level ones also add some useful side supervision. Despite the ability of MARN and MFN to learn complex representations, the simpler GRU-based Token encoder retrieves the best results at all the levels and provides more than 12% of relative improvement over the Average Embedding based model at the video level. This behavior reveals that the high complexity of MARN and MFN makes them hard to train in the context of hierarchical models with limited data leading to suboptimal performance against simpler ones such as GRU. We fix the best architecture obtained in this experiment displayed in Figure 2 and reuse it in the subsequent experiments.

6.2 Experiment 2 : What is the best strategy to take into account multiple levels of opinion information ?

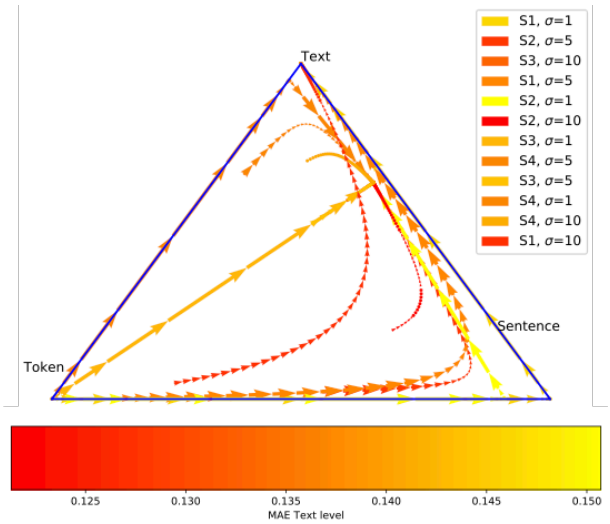


FIGURE 3 – Path of the weight vector in the simplex triangle for the different tested strategies

Motivated by the issues concerning the training of multitask losses raised in Section 4, we implemented the 4 strategies described and chose the final stationary values as the best one obtained in Experiment 1 : $(\lambda^{(N)}_{Token}, \lambda^{(N)}_{Sentence}, \lambda^{(N)}_{Text}) = (0.05, 0.5, 1)$ Note that each strategy corresponds to a path of the vector $(\lambda_{Tok}, \lambda_{Sent}, \lambda_{Tex})^T / \sum_t \lambda_t$ in the 3 dimensional simplex. We represent the 3 strategies tested in the Figure 3 corresponding to the projection of the weight vector onto the hyperplane containing the simplex.

The best paths for optimizing the *text*-level objectives are the one that smoothly move from a combination of *sentence* and *token*-level objectives to a *text* oriented one. The path in the simplex seems to be more important than the nature of the strategy since S1 and S2 reach the same *text*-level MAE score while working differently. It also appears than an objective with low σ [1] values corresponding to harder transitions tends to obtain lower scores than smooth transition based strategies. All the strategies are displayed as a function of the number of epochs in the supplementary material. In this last section we deal with the issue of the joint prediction of entities and polarities.

1. described in Section 4

	$\lambda_{Tok} = \lambda_{Sent} = 0$: no fine grained supervision						
Metric \ Model	GRU	Ind GRU	Ind GRU + att Sent	Ind GRU + att	MARN	MFN	Av Emb
MAE *Text*	0.35	0.40	0.40	0.38	0.29	0.32	**0.17**
	Supervision at the token, sentence and review levels						
Metric \ Model	GRU	Ind GRU	Ind GRU + att Sent	Ind GRU + att	MARN	MFN	Av Emb
$\mu F1$ *Tokens*	0.90	**0.93**	**0.93**	**0.93**	0.90	0.89	X
$\mu F1$ *Sentence*	0.68	0.72	**0.75**	**0.75**	0.52	0.47	X
MAE *Text*	0.16	0.15	0.15	**0.14**	0.35	0.37	X

TABLE 1 – Scores on sentiment label

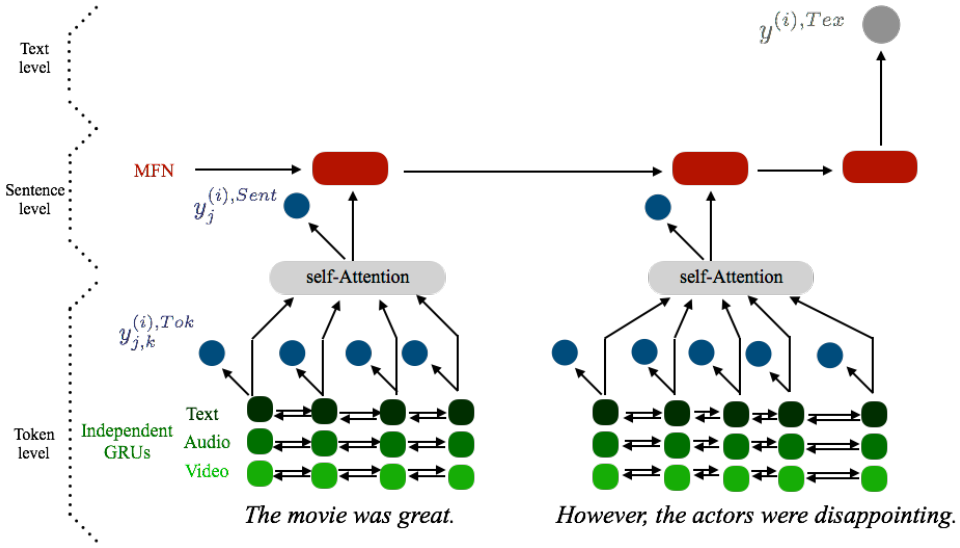


FIGURE 2 – Best architecture selected during the Experiment 1

6.3 Experiment 3 : Is it better to jointly predict opinions and entities ?

In this section, we introduce the problem of predicting the entities of the movie on which the predictions are expressed, as well as the tokens that mention them. This task is harder than the previously studied polarity prediction task due to (1) the problem of label imbalance appearing in the label distribution reported in the Table 3 and (2) the diversity of the vocabulary incurred when dealing with many entities. However since the presence of a polarity implies the presence of at least one entity, we expect that a joint prediction will perform better than an entitiy-based predictor only. Table 2 contains the results obtained with the architecture described in Figure 2 on the task of joint polarity and entity prediction as well as the results obtained when dealing with these tasks independently.

Using either the joint or the independent models provides the same results on the polarity prediction problems at the *token* and *sentence*-level. The reason is that the polarity prediction problem is easier and relying on the entities prediction would only introduce some noise in the prediction. We detail the case of *Entities* in the Table 3 and present the results obtained for the most common entity categories (among 11). As expected, the entity prediction tasks benefits from the polarity information on most of the categories except for the *Vision and special effects*. A 5% of relative improvement can be noted on the two most present *Entities* : *Overall* and *Screenplay*.

7 Conclusion

The proposed framework enables the joint prediction of the different components of an opinion based on a hierarchical neural network. The resulting models can be fully

	Polarity labels	Entity labels	Polarity + entities
F1 polarity tokens	0.93	X	0.93
F1 polarity valence	0.75	X	0.75
F1 entities tokens	X	0.97	0.97
F1 entities Entities	X	Table 3	Table 3
MAE score review level	0.14	0.38	0.14

TABLE 2 – Joint and independent prediction of entities and polarities

	Entity	Entity + Polarity	Value Count
Overall	0.71	**0.73**	1985
Actors	**0.65**	**0.65**	493
Screenplay	0.60	**0.63**	246
Atmosphere and mood	0.62	**0.64**	151
Vision and special effects	**0.62**	0.58	154

TABLE 3 – F1 score per label for the top entity categories annotated at the sentence level (mean score averaged over 7 runs), value counts are provided on the test set.

or partially supervised and take advantage of the information provided by different views of the opinions. We have experimentally shown that a good learning strategy should first rely on the easy tasks (*i.e.* for which the labels do not require a complex transformation of the inputs) and then move to more abstract tasks by benefiting from the low level knowledge. Future work will explore the use of *structured output learning* methods dedicated to the opinion structure.

A Appendix

A.1 Preprocessing details

• *Matching features and annotations :* In all our experiments we reused the descriptors presented originally in [PSC+14] and made available in the CMU-Multimodal SDK. The annotation campaign performed in [GEDBC19] had been run on the unprocessed transcripts of the spoken reviews. In order to match the setting described in previous work, we transposed the fine grained annotations from the unprocessed dataset to the processed one in the following way : We first computed the Levenstein distance (minimum number of insertion/deletion/replacement needed to transform a sequence of items into another) between the sequence of Tokens of the processed and unprocessed transcripts. Then we applied the sequence of transformations minimizing this distance on the sequence of annotation tags to build the equivalent sequence of annotation on the processed dataset.

• *Long sentences treatment :* We first removed the punctuation (denoted by the 'sp' token in the provided featurized dataset) in order to limit the maximal sentence length in the dataset. For the remaining sentences exceding 50 tokens we also applied the following treatment : We ran the sentence splitter from the spaCy library. The resulting subsentences are then kept each time they are composed of more than 4 tokens (overwise the groups of 4 tokens were merged with the next subsentence).

• *Input features cliping :* The provided feature alignment code retrieved some infinite values and impossible assignments. We clipped the values to the range [-30,30] and replaced impossible assignments by 0.

• *Training, validation and test folds :* We used the original standard folds available at : `https://github.com/A2Zadeh/CMU-MultimodalSDK/blob/master/mmsdk/mmdatasdk/dataset/standard_datasets/POM/pom_std_folds.py`

A.2 Architecture details

In this section we detail the structure of the different architectures tested in Experiment 1. According to the notations of the paper, we detail especially how the hidden representations $h^{(i),\text{Tex}}, h_j^{(i),Sent}, h_{j,k}^{(i),\text{Tok}}$ are computed in practice.

A.2.1 *token*-level model

• GRU based models

The hidden state of a Gated recurrent unit at time t : h_t^j is computed based on the previous state h_{t-1}^j and a new candidate state $\tilde{h}_{t-1}^j$:

$$h_t^j = (1 - z_j^t)h_{t-1}^j + z_t^j \tilde{h}_{t-1}^j$$

Where z_t^j is an update vector controlling how much the state is updated :

$$z_j^t = \sigma(W_z \mathbf{x}^t + U_z \mathbf{h}_{t-1})^j$$

The candidate state is computed by a simple recurrent unit with an additional reset gate $\mathbf{r}_t$:

$$\tilde{h}_t^j = (\tanh(W\mathbf{x}_t + U(\mathbf{r}_t) \odot \mathbf{h}_{t-1}))^j$$

$\odot$ is the element wise product and $\mathbf{r}_t$ is defined by :

$$r_t^j = \sigma(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1})^j$$

.

— In the case of the GRU model of Table 1, the input objects $\mathbf{x}_t$ are the concatenation of the 3 feature representations : $\mathbf{x}_t = x_t^{\text{textual}} \oplus x_t^{\text{audio}} \oplus x_t^{\text{visual}}$

— In the case of the Ind GRU Model, 3 GRU recurrent models are trained independently on each input modality and the hidden representation shared with the next parts of the network is the concatenation of the 3 hidden states : $\mathbf{h}_t = h_t^{\text{textual}} \oplus h_t^{\text{audio}} \oplus h_t^{\text{visual}}$

For these models, an entire sentence is encoded thanks to the state computed at the last token of the sentence. In the case of the GRU Ind + Att model, an additional attention model is used : it first computes a score per token u_t^j indicating its relative contribution :

$$u_t^j = \tanh(W_w h_t^j + b_w)$$

These scores are then rescaled as a probability distribution over the entire sequence :

$$\alpha_t = \frac{\mathbf{h}_t^T \mathbf{u}_t}{\sum_{t_j} \mathbf{h}_{t_j}^T \mathbf{u}_{t_j}}$$

These weights are then used to pool the hidden state representations of the sequence in a fixed length vector :

$$\mathbf{h}_{\text{Pool}} = \sum_t \alpha_t \mathbf{h}_t$$

This last representation is then used to feed the sentence level recurrent model (here a Memory fusion network). Note that the attention model does not erase the information about the modality nature of each component of $\mathbf{h}_{\text{Pool}}$ so that it can be used with a model taking into account this nature.

• MARN model

The Multi-Attention Recurrent Network from [ZLP+18] relies on 2 components :

— The Long Short Term Hybrid Memory is a LSTM model where the hidden state is the concatenation of a local hidden state (computed using the classical LSTM layer) and an external hidden state computed using a Multi Attention Block (MAB).

— The MAB block computes the external hidden state by computing 3 weighted sum of the input hidden states (one set of attention weights is computed per modality) which are then passed in 3 feedforward networks. The outputs of these network are concatenated and then passed in a second network to produce the final joint hidden representation. The detailed equations can be found in the original paper.

Similarly to the original paper we use the last hidden state computed from an entire sequence to represent it.

• MFN model

The Memory Fusion Network [ZLM+18] is made of 3 blocks :

— Each modality based sequence of feature is represented by the hidden state of a LSTM model. These hidden state are fed in the next part of the model :

— A delta attention memory takes the concatenation of two consecutive input vectors (taken from the sequence of hidden representations of the LSTM) which are fed to a feedforward model to compute an attention score for each component of these inputs. The name delta memory is only indicating the fact that the inputs are taken by pairs of inputs.

— The output of the attention layer is then sent to a Multiview Gated Memory generalizing the GRU layer to multiview data by taking into account a modality specific and a cross modality hidden representations.

The MFN model is our common model at the *sentence*-level.

A.3 Hyperparameters

All the hyper-parameters have been optimized on the validation set using MAE score at text level. Architecture optimization has been done using a random search with 15 trials. We used Adam optimizer [KB14] with a learning rate of 0.01, which is updated using a scheduler with a patience of 20 epochs and a decrease rate of 0.5 (one scheduler per classifier and per encoder). The gradient norm is clipped to 5.0, weight decay is set to 1e-5, and dropout [SHK+14] is set to 0.2. Models have been implemented in PyTorch and they have been trained on a single NVIDIA P100. The best performing MFN has a 4 attentions, the cellule size for the video is set to 48, for the audio to 32, for the text to 64. Memory dimension is set to 32, windows dimension to 2, hidden size of first attention is set to 32, hidden size of second attention is set to 16, gamma1 is set to 64, gamma2 is set to 32 [2].

A.4 Experiment 2 : Strategies displayed

In this section we report the detailed expression of the $\lambda^{(n_{\text{epoch}})}$ displayed in the figure 3.

2. For exact meaning of each parameter please refer to the official implementation which can be found here : `https://github.com/pliang279/MFN` and in the work of [ZLM+18]

A.4.1 Strategy 1

In the strategy 1, the task losses are activated one at a time following the equations :

$$\lambda_{\text{Token}}^{\text{nepoch}} = 1 - \frac{\exp\left((n_{\text{epoch}} - Ns_{\text{Token}})/\sigma\right)}{1 + \exp\left((n_{\text{epoch}} - Ns_{\text{Token}})/\sigma\right)}$$

$$\lambda_{\text{Sentence}}^{\text{nepoch}} = \frac{\exp\left((n_{\text{epoch}} - Ns_{\text{Token}})/\sigma\right)}{1 + \exp\left((n_{\text{epoch}} - Ns_{\text{Token}})/\sigma\right)} - \frac{\exp\left((n_{\text{epoch}} - Ns_{\text{Sentence}})/\sigma\right)}{1 + \exp\left((n_{\text{epoch}} - Ns_{\text{Sentence}})/\sigma\right)}$$

$$\lambda_{\text{Text}}^{\text{nepoch}} = \frac{\exp\left((n_{\text{epoch}} - Ns_{\text{Sentence}})/\sigma\right)}{1 + \exp\left((n_{\text{epoch}} - Ns_{\text{Sentence}})/\sigma\right)}$$

A.4.2 Strategy 2

In the strategy 2, the task losses are sequentially activated and maintained following the equations :

$$\lambda_{\text{Token}}^{\text{nepoch}} = 0.05$$

$$\lambda_{\text{Sentence}}^{\text{nepoch}} = 0.5\frac{\exp\left((n_{\text{epoch}} - Ns_{\text{Token}})/\sigma\right)}{1 + \exp\left((n_{\text{epoch}} - Ns_{\text{Token}})/\sigma\right)}$$

$$\lambda_{\text{Text}}^{\text{nepoch}} = \frac{\exp\left((n_{\text{epoch}} - Ns_{\text{Sentence}})/\sigma\right)}{1 + \exp\left((n_{\text{epoch}} - Ns_{\text{Sentence}})/\sigma\right)}$$

A.4.3 Strategy 3

In the strategy 3, the *Sentence* and *Text* losses are activated at the same time :

$$\lambda_{\text{Token}}^{\text{nepoch}} = 0.05$$

$$\lambda_{\text{Sentence}}^{\text{nepoch}} = 0.5\frac{\exp\left((n_{\text{epoch}} - Ns_{\text{Token}})/\sigma\right)}{1 + \exp\left((n_{\text{epoch}} - Ns_{\text{Token}})/\sigma\right)}$$

$$\lambda_{\text{Text}}^{\text{nepoch}} = \frac{\exp\left((n_{\text{epoch}} - Ns_{\text{Token}})/\sigma\right)}{1 + \exp\left((n_{\text{epoch}} - Ns_{\text{Token}})/\sigma\right)}$$

Références

[AEHP03] Salvatore Attardo, Jodi Eisterhold, Jennifer Hay, and Isabella Poggi. Multimodal markers of irony and sarcasm. *Humor*, 16(2) :243–260, 2003.

[AEP07] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in neural information processing systems*, pages 41–48, 2007.

[AHB+18] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086, 2018.

[BCE19] A. Ben Youssef, C. Clavel, and S. Essid. Early detection of user engagement breakdown in spontaneous human-humanoid interaction. *IEEE Transactions on Affective Computing*, pages 1–1, 2019.

[CC16] Chloe Clavel and Zoraida Callejas. Sentiment analysis : from opinion mining to human-agent interaction. *IEEE Transactions on affective computing*, 7(1) :74–93, 2016.

[CVMG+14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv :1406.1078*, 2014.

[GEDBC19] Alexandre Garcia, Slim Essid, Florence D'alché-Buc, and Chloé Clavel. A multimodal movie review corpus for fine-grained opinion mining. *arxiv Preprint : arXiv :1902.10102*, 2019.

[HFV+18] Leo Hemamou, Ghazi Felhi, Vincent Vandenbussche, Jean-Claude Martin, and Chloé Clavel. Hirenet : a hierarchical attention model for the automatic analysis of asynchronous video job interviews. In *AAAI 2019*. ACM, 2018.

[HL04] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[KB14] Diederik P Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

[LC15a] Caroline Langlet and Chloé Clavel. Adapting sentiment analysis to face-to-face human-agent interactions : from the detection to the evaluation issues. In *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*, pages 14–20. IEEE, 2015.

[LC15b] Caroline Langlet and Chloé Clavel. Improving social relationships in face-to-face human-agent interactions : when the agent wants to know user's likes and dislikes. In *Proceedings of the 53rd Annual Meeting of*

the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers), volume 1, pages 1064–1073, 2015.

[LC16] Caroline Langlet and Chloé Clavel. Grounding the detection of the user's likes and dislikes on the topic structure of human-agent interactions. *Knowledge-Based Systems*, 106 :116–124, 2016.

[Liu12] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1) :1–167, 2012.

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv :1508.04025*, 2015.

[MDP+11] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics : Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[MW13] James R Martin and Peter R White. *The language of evaluation*, volume 2. Springer, 2013.

[NGK+16] Behnaz Nojavanasghari, Deepak Gopinath, Jayanth Koushik, Tadas Baltrušaitis, and Louis-Philippe Morency. Deep multimodal fusion for persuasiveness prediction. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 284–288. ACM, 2016.

[NZZ17] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner : A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[PGP+16] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. Semeval-2016 task 5 : Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 19–30, 2016.

[PSC+14] Sunghyun Park, Han Suk Shim, Moitreya Chatterjee, Kenji Sagae, and Louis-Philippe Morency. Computational analysis of persuasiveness in social multimedia : A novel dataset and multimodal prediction approach. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 50–57. ACM, 2014.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove : Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[Rud17] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv :1706.05098*, 2017.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1) :1929–1958, 2014.

[SPW+13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[SWR18] Victor Sanh, Thomas Wolf, and Sebastian Ruder. A hierarchical multi-task approach for learning embeddings from semantic tasks. *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019)*, 2018.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[WWC05] Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3) :165–210, 2005.

[YYD+16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document

classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 1480–1489, 2016.

[ZLM+18] Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Memory fusion network for multi-view sequential learning. *arXiv preprint arXiv :1802.00927*, 2018.

[ZLP+18] A Zadeh, PP Liang, S Poria, P Vij, E Cambria, and LP Morency. Multi-attention recurrent network for human communication comprehension. In *AAAI*, 2018.

[ZZPM16a] Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. Mosi : multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. *arXiv preprint arXiv :1606.06259*, 2016.

[ZZPM16b] Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. Multimodal sentiment intensity analysis in videos : Facial gestures and verbal messages. *IEEE Intelligent Systems*, 31(6) :82–88, 2016.

Une Étude Empirique de la Capacité de Généralisation des Plongements de Mots Contextuels en Extraction d'Entités

Bruno Taillé[1,2], Vincent Guigue[2], et Patrick Gallinari[2]

[1]BNP Paribas, CIB, Analytics Consulting
[2]Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6

27 mai 2019

Résumé

Les plongements de mots contextuels utilisent la capacité des modèles de langue à tirer parti de données non annotées pour apprendre des représentations de mots dépendant de leur contexte. Ils sont utiles à la généralisation, particulièrement en Reconnaissance d'Entités Nommées où détecter des mentions d'entités jamais rencontrées pendant l'entraînement est crucial. Malheureusement, les benchmarks actuels surestiment l'importance des caractéristiques lexicales par rapport aux caractéristiques contextuelles à cause d'un recoupement lexical non raisonnable entre mentions d'entraînement et d'évaluation. Dans cet article, nous proposons de mieux évaluer la capacité de généralisation des modèles en séparant les mentions par degré de nouveauté ainsi qu'avec une évaluation extra domaine. Nous montrons que les plongements contextuels sont surtout bénéfiques pour la détection des mentions non rencontrées pendant l'entraînement et mènent à une amélioration relative de +9% du score micro-F1 extra domaine contre +2% intra domaine.

Mots-clef : Reconnaissance d'Entités Nommées, Plongements Contextuels, Adaptation de Domaine.

1 Introduction

La Reconnaissance d'Entités Nommées (REN) consiste à détecter les mentions textuelles d'entités et à les classifier selon des types prédéfinies. Cette tâche est modélisée comme de l'étiquetage de séquence dont l'architecture neuronale classique est le BiLSTM-CRF [HXY15]. Les progrès récents de l'état de l'art proviennent principalement de l'utilisation de nouvelles formes de représentations : des plongements de mots appris à l'échelle des caractères [LBS+16] ou des plongements de mots contextuels obtenus par des modèles de langues à l'échelle des mots [PNI+18, DCLT18] ou des caractères [ABV18].

Cette dernière approche d'apprentissage par transfert utilise des représentations de mots apprises par des modèles de langue pour réduire la dépendance aux données annotées spécifiques à une tâche ou un domaine [HR18, RSN+18]. BERT [DCLT18] établit même l'état de l'art actuel avec une simple projection linéaire des états cachés appris par un modèle de langue affiné pour la tâche. Parallèlement, l'adaptation de domaine en REN est souvent limitée au pré-entraînement d'un modèle sur des données sources dont les prédictions sont utilisées comme entrées d'un second réseau ou bien qui est réentrainé sur les données cibles [LDS18, RCL18].

Dans cet article, nous montrons que les jeux de données CoNLL03 et OntoNotes découragent la généralisation aux entités non rencontrées à cause d'un recoupement lexical non réaliste entre mentions d'entraînement et d'évaluation. Cela conduit à surestimer l'importance des caractéristiques lexicales par rapport aux caractéristiques contextuelles. Aussi, nous proposons de mieux évaluer les capacités de généralisation d'une part en séparant les mentions par degré de nouveauté et d'autre part avec une évaluation extra domaine. Dans ce cadre, nous montrons que les plongements de mots contextuels sont surtout bénéfiques pour la détection des mentions non rencontrées pendant l'entraînement et mènent à une amélioration maximale relative de +9% du score micro-F1 extra domaine contre +2% intra domaine sur CoNLL03. Cela permet d'établir une borne inférieure simple mais efficace d'adaptation de domaine sans données cibles qui pourrait être améliorée en incorporant ces dernières.

	CoNLL03					ON	ON réaligné					ON réaligné / CoNLL03				
	LOC	MISC	ORG	PER	Tous	Tous	LOC	MISC	ORG	PER	Tous	LOC	MISC	ORG	PER	Tous
Exact	82%	67%	54%	14%	52%	67%	87%	93%	54%	49%	69%	70%	78%	18%	16%	42%
Partiel	4%	11%	17%	43%	20%	24%	6%	2%	32%	36%	20%	7%	10%	45%	46%	28%
Nul	14%	22%	29%	43%	28%	9%	7%	5%	14%	15%	11%	23%	12%	38%	38%	30%

TABLE 1 – Recoupement lexical des occurrences de mentions des jeux de test avec les jeux d'entraînement respectifs pour CoNLL03 et OntoNotes original et réaligné. La dernière colonne montre le recoupement entre le test de OntoNotes réaligné et l'entraînement de CoNLL03 dans l'évaluation extra domaine.

2 Jeux de données

CoNLL03 La partie anglaise de CoNLL03 [SDM03] est le benchmark standard en REN et est composé d'articles Reuters datés de 1996 et annotés pour quatre types : Organisation (ORG), Personne (PER), Localité (LOC) et Divers (MISC).

OntoNotes 5.0 OntoNotes 5.0 [WPM+13] est composé de documents de six domaines annotés pour la REN et la Résolution de Coréférence. Il est annoté manuellement pour onze types d'entités et sept types de valeurs qui sont généralement traités sans distinction. La partition entraînement/test classique pour la REN [SVBM17] est la même que celle de la tâche de Résolution de Coréférence de CoNLL-2012 [PMX+12].

3 Recoupement Lexical

Les modèles de REN neuronaux reposent sur des caractéristiques lexicales sous la forme de plongements de mots, qu'ils soient appris au niveau des caractères ou non. Bien que la syntaxe soit incorporée par leur pré-entraînement non supervisé ou l'architecture du réseau, nous prétendons que CoNLL03 et OntoNotes évaluent mal la capacité de généralisation des algorithmes à cause d'un important recoupement lexical entre les mentions présentent dans le jeu d'entraînement et les jeux de validation et de test. Nous le quantifions en séparant les occurrences des mentions dans les sets d'évaluation en trois catégories : recoupement exact, recoupement partiel et recoupement nul, de manière similaire à Augenstein et al. [ADB17].

Une mention d'un jeu d'évaluation est un recoupement exact si elle apparaît sous l'exacte même forme sensible à la capitalisation dans le jeu d'entraînement et annotée avec le même type. Le recoupement est partiel s'il n'est pas exact mais qu'au moins un des mots non vides de la mention apparaît dans une mention de même type. Toutes les autres mentions ont un recoupement nul : leurs mots non vides ne sont jamais rencontrés pendant l'entraînement. Ainsi, la proportion de recoupements partiels et nuls reflète la capacité d'un jeu de test à évaluer la capacité de généralisation d'un algorithme aux entités non rencontrées, ce qui est un premier pas nécessaire à l'adaptation de domaine.

Comme reporté dans la Table 1, les deux jeux de données montrent un important recoupement lexical de mentions. Dans CoNLL03, plus de la moitié des occurrences de mentions du jeu de test est présente dans le jeu d'entraînement alors que seulement 28% sont totalement nouvelles. Dans OntoNotes, le recoupement est encore pire avec 67% de recoupement exact contre 9% de nouvelles mentions. De plus, nous remarquons une influence significative du type d'entité puisque LOC et MISC présentent le recoupement le plus important alors que PER et ORG ont un vocabulaire plus varié.

Cela montre que les deux principaux jeux de données étalons en REN en anglais évaluent surtout la performance d'extraction des mentions déjà rencontrées lors de l'entraînement, bien qu'apparaissant dans des phrases différentes. De telles proportions de recoupement lexical ne sont pas réalistes dans des applications réelles où un modèle doit traiter quelques ordres de grandeurs de documents de plus en inférence qu'en entraînement pour rentabiliser le coût de l'annotation. L'amélioration spécifique des performances sur les nouvelles mentions revêt donc une importance cruciale dans un cas concret qui est sous-estimée par les benchmarks actuels.

Nous proposons donc une évaluation extra domaine en entraînant les modèles sur CoNLL03 et en les testant sur OntoNotes, plus grand et plus diversifié, ce qui correspond mieux au cas concret. Nous gardons les types de CoNLL03 et y alignons ceux d'OntoNotes : ORG et PER correspondent déjà et nous alignons LOC + GPE dans OntoNotes à LOC dans CoNLL et NORP + LANGUAGE à MISC. Cela réduit le recoupement exact à 42%, ce qui nous semble encore une surestimation du recoupement en utilisation réelle.

				CoNLL03				OntoNotes			
Entraînement	Modèle	Représentation	Dim	Exact	Partiel	Nul	Tous	Exact	Partiel	Nul	Tous
CoNLL03	**BiLSTM-CRF**	BERT	4096	95.7	88.8	82.2	90.5	95.1	82.9	73.5	**85.0**
		ELMo	1024	95.9	89.2	85.8	**91.8**	94.3	79.2	72.4	83.4
		Flair	4096	95.4	88.1	83.5	90.6	94.0	76.1	62.1	79.0
		GloVe + char	350	95.3	85.5	83.1	89.9	93.9	73.9	60.4	77.9
		GloVe	300	95.1	85.3	81.1	89.3	93.7	73.0	57.4	76.9
	Map-CRF	BERT	4096	93.2	85.8	73.7	86.2	93.5	77.8	67.8	80.9
		ELMo	1024	93.7	87.2	80.1	**88.7**	93.6	79.1	69.5	**82.2**
		Flair	4096	94.3	85.1	78.6	88.1	93.2	74.0	59.6	77.5
		GloVe + char	350	93.1	80.7	69.8	84.4	91.8	69.3	55.6	74.8
		GloVe	300	92.2	77.0	61.7	81.5	89.6	62.8	38.5	68.1
OntoNotes	**BiLSTM-CRF**	BERT	4096					96.9	88.6	81.1	**93.5**
		ELMo	1024					97.1	88.0	79.9	93.4
		Flair	4096					96.7	85.8	75.0	92.1
		GloVe + char	350					96.3	83.3	69.9	91.0
		GloVe	300					96.2	82.9	63.8	90.4

TABLE 2 – Scores micro-F1 séparés par degré de recoupement en évaluation intra et extra domaine. Nos résultats sont obtenus en moyennant cinq entraînements.

4 Représentations de Mots

Plongements de mots classiques Nous prenons **GloVe** [PSM14] comme base de référence des plongements traditionnels. Bien que les plongements GloVe soient calculés sur un corpus important pour capturer une similarité sémantique basée sur la co-occurrence, cette représentation est purement lexicale puisque chaque mot est aligné à une unique représentation. Les plongements sont initialisés avec GloVe 840B et leurs valeurs sont affinées pendant l'entraînement.

Plongements de mots à l'échelle des caractères Nous reproduisons le **Char-BiLSTM** de Lample et al. [LBS+16], un BiLSTM au niveau de chaque mot qui apprend sa représentation à partir des plongements de ses caractères pour tenir compte de caractéristiques orthographiques et morphologiques. Le Char-BiLSTM est entraîné conjointement au réseau de REN et ses sorties sont concaténées aux plongements GloVe.

Plongements de mots contextuels Contrairement aux représentations précédentes, les plongements de mots contextuels prennent en compte le contexte d'un mot dans sa représentation. Pour se faire, un modèle de langue est préentrainé sur un corpus non annoté et on prend sa représentation interne de la prédiction d'un mot sachant son contexte. **ELMo** [PNI+18] utilise un réseau convolutif à l'échelle des caractères (Char-CNN) pour obtenir un plongement de mot indépendant du contexte et la concaténation de modèles de langue LSTM à deux couches en sens avant et inverse pour la contextualisation. **BERT** [DCLT18] adopte des plongements de sous-mots et apprend une représentation dépendant des contextes droits et gauches en entraînant l'encodeur d'un Transformer [VSP+17] pour un modèle de langue masqué et la prédiction de la phrase suivante. Nous utilisons le modèle "$\text{BERT}_{\text{LARGE}}$ feature-based" pour une comparaison plus juste : les poids du modèle de langue sont gelés et nous concaténons les états cachés de ses quatre dernières couches. **Flair** [ABV18] emploie directement un modèle de langue à l'échelle du caractère. Comme pour ELMo, deux modèles de langue LSTM de sens opposés sont entraînés et leurs sorties concaténées. Flair et ELMo sont pré-entraînés sur le 1 Billion Word Benchmark [CMS+13] alors que BERT l'est sur la réunion de Book Corpus [ZKZ+15] et Wikipedia en anglais.

5 Expériences

5.1 Cadre Expérimental

Pour effectuer la REN, nous plaçons les représentations de mots dans deux modèles : un BiLSTM-CRF [HXY15] avec une dimension cachée de 100 dans chaque direction et Map-CRF pour lequel elles sont projetées linéairement dans l'espace de sortie. Nous gardons le CRF [LMP01] car la projection de plongements non contextuels revient à une prédiction indépendante pour chaque mot.

Nous séparons les Précision, Rappel et score F1 par degré de recoupement exact, partiel ou nul. Pour la Précision, cette séparation est effectuée a posteriori sur les prédictions du modèle. Nous utilisons le schéma

d'annotations IOBES et validons sur le score-micro F1. Nous rapportons les moyennes des scores obtenus par 5 entraînements différents. Pour chaque modèle, nous choisissons le meilleur de SGD ou Adam avec un taux d'apprentissage de 0.001, des batchs de taille 64, un dropout de 0.5 et un early stopping avec patience 5.

Les scores F1 intra et extra domaine des modèles entraînés sur CoNLL03 sont rapportés dans la Table 2 ainsi que les bornes supérieures extra domaine obtenues en entraînement sur OntoNotes réaligné. Nous omettons délibérément l'évaluation extra domaine de OntoNotes vers CoNLL03 en considérant que les cas d'application concrets sont toujours limités en ressources annotées et ainsi entraîné sur le jeu de données le plus petit et le moins varié.

5.2 Résultats

Performances Intra Domaine Tout d'abord, dans toutes les configurations le score F1 est le plus haut pour les recoupements exacts, puis partiels et nuls ce qui confirme le biais dans les jeux de données avec un recoupement lexical important. Ensuite, bien que ELMo apparaît comme la solution la plus stable intra domaine, il est difficile de dégager une hiérarchie claire entre plongements contextuels puisque les données de pré-entraînement ainsi que la dimension des représentations diffèrent. Pour BERT et Flair, le BiLSTM-CRF performe relativement moins bien sur CoNLL03 que sur OntoNotes, probablement par sur-apprentissage sur CoNLL03. De plus, le gain maximal de la contextualisation sur CoNLL03 est de +0.6 F1 en recoupement exact contre +3.7 en partiel et +2.7 en nul. D'autre part, Map-CRF avec ELMo ou Flair arrive presque au même niveau que BiLSTM-CRF et Glove + char, ce qui montre que les modèles de langues capturent intrinsèquement des représentations utiles à la REN. Enfin, quelle que soit la représentation le BiLSTM réduit l'écart de performance entre mentions vues et non vues.

Généralisation Extra Domaine En évaluation extra domaine, les performances se dégradent et l'écart se creuse entre les mentions vues et non vues. De plus, la contextualisation est encore plus bénéfique aux mentions non vues avec +1.2 F1 en recoupement exact, +9.0 en partiel et +13.1 en nul avec le BiLSTM-CRF et BERT. Cette amélioration provient clairement du pré-entraînement du modèle de langue puisque même avec Map-CRF, les plongements contextuels atteignent au moins 77.5 F1 contre 77.9 pour BiLSTM-CRF et GloVe + char. Nous distinguons néanmoins une séparation entre plongements contextuels puisque Flair, issu d'un modèle de langue à l'échelle des caractères, généralise moins bien que ELMo ou BERT en extra domaine pour les deux modèles. Il ressort ainsi que contextualiser des mots ou sous-mots conduit à une meilleure généralisation en REN. Enfin, nous pouvons séparer les performances par genres des documents dans OntoNotes comme rapporté dans la Table 3. Pour tous les modèles, la meilleure adaptation se fait pour le type *broadcast news* qui est plus proche du domaine de CoNLL03 que *web text* ou *magazine*. Cependant, les plongements contextuels bénéficient principalement aux genres plus distants et mènent à des résultats plus homogènes.

	bc	bn	nw	mz	tc	wb	Tous
BERT	87.2	88.4	84.7	82.4	84.5	79.5	**85.0**
ELMo	85.0	88.6	82.9	78.1	84.0	79.9	83.4
Flair	78.0	86.5	80.4	71.1	73.5	72.1	79.0
GloVe + char	80.4	86.3	77.0	70.7	79.7	69.2	77.9

Table 3 – Scores micro-F1 extra domaine du BiLSTM-CRF par genres. Respectivement *broadcast conversation*, *broadcast news*, *news wire*, *magazine*, *telephone conversation* et *web text*.

Influence du Type Bien que le score micro-F1 soit souvent la seule métrique rapportée en REN, les types d'entités devraient être pris en compte. Comme montré dans la Table 4, en intra domaine MISC est le type le plus difficile à détecter alors que PER est le plus facile, certainement grâce à un motif prénom-nom fréquent. Cependant, la contextualisation bénéficie homogènement à tous les types en intra domaine alors qu'elle bénéficie surtout à ORG et PER en extra domaine. Cela s'explique par moins de 18% de recoupement exact avec le set d'entraînement contre plus de 70% pour LOC et MISC. Ainsi, la contextualisation est plus utile pour la généralisation aux types avec le plus de variation lexicale même quand ils sont plus faciles à détecter en intra domaine.

6 Travaux Connexes

Augenstein et al. [ADB17] présentent une étude quantitative de deux modèles basés sur les CRF et un réseau convolutif avec des plongements de mots classiques [CW11] sur sept jeux de données dont CoNLL03 et OntoNotes. Ils séparent notamment les performances sur les mentions rencontrées en entraînement (notre recoupement exact) de celles non rencontrées et montrent une chute du score F1 sur ces dernières.

	LOC				MISC				ORG				PER				Tous
	Exact	Partiel	Nul	Tous	Exact	Partiel	Nul	Tous	Exact	Partiel	Nul	Tous	Exact	Partiel	Nul	Tous	Tous
CoNLL03 → CoNLL03																	
BERT	96.1	68.7	76.5	92.1	93.6	53.5	46.9	79.7	95.2	82.0	79.1	88.0	99.0	98.1	93.2	96.2	90.5
ELMo	96.0	72.6	83.3	93.1	94.3	58.9	53.9	81.8	96.0	80.8	83.5	89.6	98.9	98.7	94.6	97.0	**91.8**
Flair	95.7	73.3	79.7	92.3	93.5	55.9	49.2	80.2	95.1	77.9	80.9	87.8	98.3	98.3	93.3	96.2	90.6
GloVe + char	95.6	64.1	80.5	91.8	93.3	54.0	40.8	78.9	94.9	74.4	82.0	87.5	98.7	97.2	92.0	95.2	89.9
CoNLL03 → OntoNotes																	
BERT	96.1	65.7	79.0	89.6	94.1	51.2	25.8	72.6	93.6	83.6	76.4	82.6	93.5	90.3	83.4	88.2	**85.0**
ELMo	94.9	63.0	77.7	88.5	94.6	56.5	37.8	78.8	92.8	80.8	74.5	80.5	91.9	84.5	75.6	82.2	83.4
Flair	95.3	59.7	67.8	86.2	94.0	52.8	28.4	74.2	89.3	77.2	59.9	72.6	92.0	82.1	70.1	78.8	79.0
GloVe + char	95.6	62.4	69.3	86.7	93.8	56.7	30.1	75.3	88.9	74.0	58.5	70.8	89.5	78.9	64.7	74.8	77.9

Table 4 – Scores F1 par type du BiLSTM-CRF entraîné sur CoNLL03 en évaluation intra et extra domaine.

Moosavi et Strube [MS17] soulèvent un phénomène similaire en Résolution de Coréférence sur CoNLL-2012 et montrent qu'en évaluation extra domaine l'écart de performance entre les modèles d'apprentissage profond et un système de règles disparaît. Dans [MS18], ils proposent d'utiliser des caractéristiques linguistiques (comme le genre, le type d'entité ou la catégorie grammaticale) pour améliorer la généralisation extra domaine. Néanmoins, de telles caractéristiques sont obtenues en utilisant des modèles à leur tour entraînés avec des caractéristiques lexicale et sur des données ou le même problème de recoupement lexical se pose, au moins pour la Reconnaissance d'Entités Nommées.

7 Conclusion

Les benchmarks actuels de REN sont donc biaisés en faveur des mentions déjà rencontrées, à l'exact opposé des applications concrètes. D'où la nécessité de séparer les performances par degré de recoupement des mentions pour mieux évaluer les capacités de généralisation. Dans ce cadre, les plongements contextuels bénéficient plus significativement aux mentions non rencontrées, d'autant plus en extra domaine.

Les travaux futurs peuvent chercher à réduire encore l'écart de performance entre mentions rencontrées ou non, améliorer les capacités d'adaptation de domaine zero-shot avec des données cibles additionnelles ou aborder la généralisation multilingue en utilisant des modèles de langues entraînés sur des corpus multilingues.

Remerciements

Nous remercions Geoffrey Scoutheeten et Victor Storchan pour leurs points de vue et commentaires précieux. Ce travail est principalement financé par BNP Paribas dans le cadre de la convention CIFRE 2018/0327.

Références

[ABV18] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[ADB17] Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. Generalisation in named entity recognition : A quantitative analysis. *Computer Speech & Language*, 44 :61–83, 7 2017.

[CMS+13] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. *arXiv preprint arXiv :1312.3005*, 2013.

[CW11] Ronan Collobert and Jason Weston. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12 :2493–2537, 2011.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv :1810.04805*, 2018.

[HR18] Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 328–339, 2018.

[HXY15] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv preprint arXiv :1508.01991*, 2015.

[LBS+16] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT 2016*, pages 260–270, 2016.

[LDS18] Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. Transfer Learning for Named-Entity Recognition with Neural Networks. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, pages 4470–4473, 2018.

[LMP01] John Lafferty, Andrew McCallum, and Fernando C N Pereira. Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning*, volume 8, pages 282–289, 2001.

[MS17] Nafise Sadat Moosavi and Michael Strube. Lexical Features in Coreference Resolution : To be Used With Caution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 14–19, 2017.

[MS18] Nafise Sadat Moosavi and Michael Strube. Using Linguistic Features to Improve the Generalization Capability of Neural Coreference Resolvers. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, 2018.

[PMX+12] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. CoNLL-2012 shared task : Modeling multilingual unrestricted coreference in OntoNotes. *Joint Conference on EMNLP and CoNLL-Shared Task. Association for Computational Linguistics*, pages 1–40, 2012.

[PNI+18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, 2 2018.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe : Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[RCL18] Juan Diego Rodriguez, Adam Caldwell, and Alexander Liu. Transfer Learning for Entity Recognition of Novel Classes. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1974–1985, 2018.

[RSN+18] Alec Radford, Tim Salimans, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. page 12, 2018.

[SDM03] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task. In *Proceedings of the seventh Conference on Natural Language Learning at NAACL-HLT 2003*, volume 4, pages 142–147, 2003.

[SVBM17] Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680, 2017.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[WPM+13] Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, and Michelle Franchini. OntoNotes Release 5.0 LDC2013T19. *Linguistic Data Consortium, Philadelphia, PA*, 2013.

[ZKZ+15] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning Books and Movies : Towards Story-like Visual Explanations by Watching Movies and Reading Books. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 19–27, 2015.

Passty Langevin

Pascal Bianchi[1], Adil Salim[2], et Sholom Schechtman[3]

[1]LTCI, Télécom ParisTech, France
[2]VCC, KAUST, Saudi Arabia
[3]LIGM, Université Paris-Est Marne-la-Vallée, France

4 juin 2019

Résumé

Dans cet article nous proposons une nouvelle méthode pour simuler selon une densité log-concave lorsque la fonction convexe sous-jacente est une composée d'une fonction convexe et un opérateur affine. Ce type de problème est courant en machine learning ou imagerie computationnelle. Notre algorithme est une variante des algorithmes dits de Langevin, l'idée principale étant, par analogie avec les méthodes d'optimisation convexe, de poser un problème équivalent plus simple qu'on résout avec une légère modification de PGLA (Proximal Gradient Langevin Algorithm). Les analyses de convergence sont faites à l'aide de nouveaux liens établis entre ces méthodes et les algorithmes d'optimisation convexe. Des résultats expérimentaux sont présentés en dernière partie.

Mots-clef : Langevin, optimisation, MCMC

1 Introduction

Dans ce papier on s'intéresse à un cas particulier du problème général qui est de simuler un échantillon distribué selon une loi de densité log-concave $\pi \propto e^{-U}$, où $U : \mathbb{R}^d \to \mathbb{R}$ est une fonction convexe réelle.

La dimension de l'espace de départ d étant en général très grande pour simuler selon π on utilise des méthodes MCMC s'inspirant de l'équation différentielle stochastique de Langevin suivante :

$$\mathrm{d}X_t = -\nabla U(X_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}B_t$$

où B_t est un mouvement brownien.

En effet, sous des conditions assez faibles sur U (cf. par exemple [RT96]) le processus solution de (1) aura comme mesure stationnaire π. Ainsi si on arrive à simuler une solution de (1) on aura asymptotiquement accès à l'échantillon voulu.

Les algorithmes de Langevin ciblant π sont alors une discrétisation sous une forme ou une autre de (1). Une des premières variantes est le ULA (Unajusted Langevin Algorithm) introduit dans [RT96] :

$$X_{k+1} = X_k - \gamma_k \nabla U(X_k) + \sqrt{2\gamma} W_{k+1} \qquad (1)$$

Avec W_{k+1} une gaussienne centrée réduite et $(\gamma_n)_{n\in\mathbb{N}}$ une suite de pas.

Des premiers résultats non-asymptotiques de la convergence (en norme TV) de la loi des itérés de ULA vers π ont été obtenus en 2013 par [Dal17]. Remarquons que la discrétisation (1) introduit un biais, et on n'a plus en général la convergence de la distribution des X_k vers la loi cible. Cependant ce biais peut être contrôlé en jouant sur la taille des pas $(\gamma_n)_{n\in\mathbb{N}}$.

Une autre remarque est le fait qu'une itération de (1) est pratiquement (à une perturbation gaussienne près) l'algorithme de descente de gradient visant à minimiser la fonction U. Ce premier lien entre l'optimisation et les méthodes de Langevin a été renforcé dans les trois papiers sortis en printemps 2018 ([DMM18], [Wib18], [Ber18]) les auteurs décomposent une itération de (1) en deux parties : en premier lieu la "descente de gradient" vise à minimiser U (ou ce qui est équivalent l'énergie potentielle associé), et le bruit gaussien cherche à minimiser l'entropie (l'entropie et l'énergie potentielle sont définis aux équations (10) et (11)). De ce fait, par analogie avec les méthodes d'optimisation des fonctions composées ULA tend à minimiser la somme des deux qui se trouve être (à constante près) la divergence de Kullback-Leibler par rapport à π.

Ce point de vue permet non seulement de mieux comprendre la nature des méthodes de Langevin, mais introduit aussi de nouvelles techniques de preuves et ouvre la voie à de nouveaux algorithmes. Ainsi dans [Ber18] l'étape de descente de gradient est remplacée

par un pas proximal, alors que dans [DMM18] le cas où U a une composante non lisse est étudié et la descente de gradient devient une descente de gradient proximale.

Le cas qui nous intéresse est le cas où U s'écrit sous une forme particulière :

$$U = f + g \circ \mathrm{M} \tag{2}$$

avec f et g deux fonctions convexes et M un opérateur affine. On supposera de plus que l'opérateur proximal de $g \circ M$ ne peut pas être calculé facilement. Des problèmes de ce type interviennent en machine learning et imagerie computationnelle dans un cadre bayésien, ici $g \circ \mathrm{M}$ représenterait la norme TV (cf. [Per16], [DMP18] pour plus de détails), une application au filtrage sur graphe sera donnée section 5

Si notre but était de trouver le minimum de U, un moyen de procéder serait alors d'effectuer le changement de variable $y = \mathrm{M}\, x, y \in \mathcal{Y}$ (ou $\mathcal{Y}$ est l'espace de départ de g) et remarquer que :

$$\inf_{x \in \mathbb{R}^d} U(x) = \inf_{y \in \operatorname{Im} \mathrm{M}} f(\mathrm{M}^{-1}\, y) + g(y) = \inf_{y \in \mathcal{Y}} F(y) + g(y)$$

où $F(y) = f(\mathrm{M}^{-1}\, y)$ si $y \in \operatorname{Im} \mathrm{M}$ et $+\infty$ sinon. Si l'opérateur proximal de F peut être calculé on peut alors trouver le minimum recherché par le schema de Passty [Pas79] :

$$y_{k+1} = \operatorname{prox}_F^{\gamma}(\operatorname{prox}_g^{\gamma}(y_k)) \tag{3}$$
$$x_{k+1} = \mathrm{M}^{-1}\, y_{k+1} \tag{4}$$

L'algorithme que nous proposons Passty-Langevin (PL) est alors analogue à ce schema. Nous minimisons dans E la fonction $f(M^{-1}\cdot)+g(\cdot)$ par la méthode qu'on vient de décrire et nous rajoutons un bruit gaussien dans l'espace E. Plus précisément en notant $(\gamma_k)_{k\in\mathbb{N}}$ une suite de pas l'algorithme que nous proposons est :

$$Y_{k+1} = \operatorname{prox}_F^{\gamma_{k+1}}(\operatorname{prox}_g^{\gamma_k}(Y_k)) + \sqrt{2\gamma_{k+1}}\Xi_{k+1} \tag{5}$$
$$X_{k+1} = \mathrm{M}^{-1}\, Y_{k+1} \tag{6}$$

où Ξ_{k+1} est une gaussienne standard dans $\operatorname{Im} M$. Nous montrerons que sous des conditions de régularité sur f et g, la convergence de la loi de X_{k+1} vers π.

Nous introduisons les concepts mathématiques nécessaires en section 2 puis posons rigoureusement le problème et présentons notre algorithme en section 3. En 4 l'analyse de convergence de l'algorithme est effectuée, la structure de cette partie, les résultats et les techniques des démonstrations suivent de près ceux de [DMM18]. Enfin en section 5 on compare notre algorithme à l'état de l'art sur le problème de filtrage sur graphe bayésien. C'est un cas où l'opérateur proximal de $g \circ \mathrm{M}$ de (2) ne peut qu'être approché par un algorithme itératif, les expériences numériques confirment alors l'avantage de notre méthode.

2 Notations

Pour $\gamma > 0$ et $\phi : \mathbb{R}^k \to \mathbb{R}$ l'opérateur proximal de pas γ pour ϕ est définit comme suit :

$$\operatorname{prox}_\phi^\gamma(x) = \arg\min_{y \in \mathbb{R}^k} \{ \frac{1}{2\gamma} \|y - x\|^2 + \phi(y) \} \tag{7}$$

On sait que pour les fonctions convexes cet opérateur est bien défini, et vérifie la propriété suivante (voir par exemple [NPB14])

$$\operatorname{prox}_\phi^\gamma(x) \in x - \gamma \partial\phi(\operatorname{prox}_\phi^\gamma(x)) \tag{8}$$

où $\partial\phi(x) = \{v \in \mathbb{R}^k / \phi(y) \geqslant \phi(x) + \langle v, y - x \rangle \ \forall y \in \mathbb{R}^k\}$ est le sous-gradient de ϕ en x.

On note $\mathcal{P}_2(\mathbb{R}^d)$ l'espace des mesures de probabilités sur $\mathbb{R}^d$ admettant un moment d'ordre deux. Pour un sous-espace vectoriel $E \subset \mathbb{R}^d$, $\mathcal{P}_2(E) \subset \mathcal{P}_2(\mathbb{R}^d)$ sont les mesures de probabilités à support dans E.

Pour tout $\mu, \nu \in \mathcal{P}_2(\mathbb{R}^d)$ on définit la distance de wasserstein d'ordre deux $\mathcal{W}_2$:

$$\mathcal{W}_2^2(\mu, \nu) = \inf_{\substack{Y \sim \nu \\ X \sim \mu}} \mathbb{E}[(X - Y)^2] \tag{9}$$

$\mathcal{P}_2(\mathbb{R}^d)$ muni de cette distance est un espace métrique, séparable, complet et l'infimum dans (9) est atteint (cf. [Vil09] chapitre 6)

Pour E un sous-espace vectoriel de $\mathbb{R}^d$ nous définissons l'entropie relative à E comme une fonctionnelle $\mathcal{H}_E : \mathcal{P}_2(\mathbb{R}^d) \to]-\infty, +\infty]$:

$$\mathcal{H}_E(\mu) = \begin{cases} \int_E \frac{\mathrm{d}\mu}{\mathrm{d}\lambda_E} \log\left(\frac{\mathrm{d}\mu}{\mathrm{d}\lambda_E}\right) \mathrm{d}\lambda_E & \text{si } \mu \ll \lambda_E \\ +\infty \text{ sinon} \end{cases} \tag{10}$$

Pour U une fonction convexe nous définissons l'énergie potentielle associée à U, $\mathcal{E}_U(\mu) : \mathcal{P}_2(\mathbb{R}^d) \to]-\infty, +\infty]$ comme :

$$\mathcal{E}_U(\mu) = \int_x U(x)\mu(\mathrm{d}x) \tag{11}$$

La somme des deux est alors la fonctionnelle d'énergie libre associée à U.

$$\mathcal{F}(\mu) = \mathcal{H}_E(\mu) + \mathcal{E}_U(\mu) \tag{12}$$

Un calcul simple montre que si $\pi_E \in \mathcal{P}_2(\mathbb{R}^d)$ admet Ce^{-U} comme densité de probabilité par rapport à λ_E on a :

$$\mathcal{F}(\mu) = \mathrm{KL}(\mu|\pi) + \log(C) \tag{13}$$

où KL est la distance de Kullback-Leibler. (13) montre entre autre que π_E minimise $\mathcal{F}$. variable

3 Position du problème

Nous nous intéressons à simuler un échantillon selon une loi de probabilité absolument continue par rapport à $\lambda_{\mathbb{R}^d}$ de densité :

$$\pi(\mathrm{d}x) = \frac{e^{-U(x)}}{\int_{\mathbb{R}^d} e^{-U(x)}\lambda_{\mathbb{R}^d}(\mathrm{d}x)}\lambda_{\mathbb{R}^d}(\mathrm{d}x) \tag{14}$$

Nous supposons que la fonction $U : \mathbb{R}^d \to \mathbb{R}$ se décompose en $U = f + g \circ M$, avec $f : \mathbb{R}^d \to \mathbb{R}$ une fonction convexe différentiable de gradient L-Lipschitz, $g : \mathbb{R}^p \to \mathbb{R}$ une fonction convexe continue C-Lipschitz. Et $M : \mathbb{R}^d \to \mathbb{R}^p$ est une fonction affine injective.

Dans la suite on note le sous-espace vectoriel $E = \operatorname{Im} \mathrm{M}$, la dimension de E est donc d. Nous notons $F : \mathbb{R}^p \to \overline{\mathbb{R}}$ l'infimale post-composition de M par f ([BC11] section 12.5) devient par injectivité de M :

$$\begin{aligned} F(y) &= \inf_{y \in \operatorname{Im} \mathrm{M}} f(\mathrm{M}^{-1}\, y) \\ &= \begin{cases} f(\mathrm{M}^{-1}\, y) \text{ si } y \in \operatorname{Im} \mathrm{M} \\ +\infty \text{ sinon} \end{cases} \end{aligned} \tag{15}$$

F est alors convexe ([BC11] proposition 12.36) de domaine E.

Avec ces notations nous notons $\pi_E \in \mathcal{P}_2(E)$ la mesure de probabilité absolument continue par rapport à λ_E :

$$\pi_E(\mathrm{d}y) = \frac{e^{-F(y)+g(y)}}{\int e^{-F(y)+g(y)}\lambda_E(\mathrm{d}y)}\lambda_E(\mathrm{d}y) \tag{16}$$

par un simple changement de variable on a que si $Y \sim \pi_E$, $M^{-1}Y \sim \pi$

Une fois placé dans E notre algorithme est simplement une variante des algorithmes de Langevin usuels. Plus précisément partant d'une variable aléatoire $Y_0 \sim \mu_0 \in \mathcal{P}_2(E)$ et d'une suite de pas $(\gamma_k)_{k\in\mathbb{N}} \in \mathbb{R}_+$ l'algorithme que nous étudions (PL) est :

$$\begin{aligned} Y_{k+1/3} &= \operatorname{prox}_g^{\gamma_k}(Y_k) \\ Y_{k+2/3} &= \operatorname{prox}_F^{\gamma_{k+1}}(Y_{k+1/3}) \\ Y_{k+1} &= Y_{k+2/3} + \sqrt{2\gamma_{k+1}}\Xi_{k+1} \\ X_{k+1} &= \mathrm{M}^{-1}\, Y_{k+1} \end{aligned} \tag{17}$$

où Ξ_{k+1} est une gaussienne standard sur E.
Nous montrerons dans la section suivante qu'asymptotiquement la loi de Y_k sera proche de π_E en KL, par invariance par transformation affine de la KL on aura alors simplement la proximité de la loi de X_k à π

4 Analyse de la convergence

Une fois qu'on est sur l'espace E, l'algorithme (17) est pratiquement similaire au PGLD (Proximal Gradient Langevin Dynamics) de [DMM18] (section 4.2), nous remplaçons juste une descente de gradient par une descente proximale et prenons quelques précautions supplémentaire dû à la forme de F. Les théorèmes et les résultats présentés dans cette section suivent donc [DMM18].

Comme le montre l'équation (13) π_E est le minimiseur de $\mathcal{F}$. Pour prouver la convergence de l'algorithme nous contrôlerons à chaque itération la quantité $\mathcal{F}(\mu_{k+1}) - \mathcal{F}(\pi_E)$. Ainsi on introduit la décomposition suivante :

$$\begin{aligned} \mathcal{F}(\mu_{k+1}) - \mathcal{F}(\pi_E) = \mathcal{H}_E(\mu_{k+1}) - \mathcal{H}_E(\pi_E) \\ + \mathcal{E}_U(\mu_{k+1}) - \mathcal{E}_U(\mu_{k+2/3}) \\ + \mathcal{E}_U(\mu_{k+2/3}) - \mathcal{E}_U(\pi_E)) \end{aligned}$$

où μ_i est la distribution de probabilité de Y_i.
Le terme impliquant l'entropie est contrôlé par des techniques de flots de gradients comme dans [DMM18], les incréments d'énergie libre sont gérés à l'aide des propriétés de régularité et convexité de g et de F.

Lemme 1. *Si f est de gradient L-Lipschitz on a :*

$$\mathcal{E}_F(\mu_{k+1}) - \mathcal{E}_F(\mu_{k+2/3}) \leqslant Lp||\,\mathrm{M}^{-1}\,||_2^2\gamma_{k+1}$$

Démonstration. Soit x, y dans E. Comme f est convexe et de gradient L-Lipschitz on a :

$$\begin{aligned} F(x+y) - F(x) &= f(\mathrm{M}^{-1}(x+y)) - f(\mathrm{M}^{-1}\, x)) \\ &\leqslant \langle \nabla f(\mathrm{M}^{-1}\, x), \mathrm{M}^{-1}\, y\rangle + \frac{L}{2}||\,\mathrm{M}^{-1}\, y||^2 \end{aligned}$$

En prenant deux variables aléatoires Y, Z telles que $Y \sim \mu_{k+2/3}$ et Z une gaussienne standard dans $\operatorname{Im} E$ on a $Y + Z \sim \mu_{k+1}$. Ainsi en passant à l'espérance on obtient :

$$\begin{aligned} &\mathcal{E}_F(\mu_{k+1}) - \mathcal{E}_F(\mu_{k+2/3}) \\ &\leqslant \frac{\int_y\int_z \langle \nabla f(\mathrm{M}^{-1}\, y), \mathrm{M}^{-1}\, z\rangle e^{-\frac{||z-y||^2}{4\gamma}}\lambda_E(\mathrm{d}z)\lambda_E(\mathrm{d}y)}{(4\pi\gamma)^{p/2}} \\ &+ \frac{\int_y\int_z \frac{L}{2}||\,\mathrm{M}^{-1}\, z||^2 e^{-\frac{||z-y||^2}{4\gamma}}\lambda_E(\mathrm{d}z)\lambda_E(\mathrm{d}y)}{(4\pi\gamma)^{p/2}} \\ &\leqslant Lp||\,\mathrm{M}^{-1}\,||_2^2\gamma \end{aligned}$$

□

Lemme 2. *$\forall \nu \in \mathcal{P}_2(E)$ on a :*

$$\begin{aligned} &2\gamma_{k+1}(\mathcal{E}_F(\mu_{k+2/3}) - \mathcal{E}_F(\nu)) \\ &\quad \leqslant \mathcal{W}_2^2(\mu_{k+1/3}, \nu) - \mathcal{W}_2^2(\mu_{k+2/3}, \nu) \end{aligned} \tag{18}$$

Démonstration. Soit x, z dans $\mathbb{R}^d$, et $y = \mathrm{prox}_F^{\gamma_{k+1}}(x)$. Alors on a :

$$\begin{aligned} ||y-z||^2 &= ||z-x||^2 + 2\langle y-x, x-z\rangle + ||y-x||^2 \\ &= ||z-x||^2 + 2\langle y-x, y-z\rangle - ||y-x||^2 \end{aligned}$$

On a d'après (8) $x - y \in \gamma_{k+1}\partial F(y)$ donc par convexité de F, on a $\langle y-x, y-z\rangle \leqslant \gamma_{k+1}(F(z) - F(y))$ et :

$$||y-z||^2 \leqslant ||z-x||^2 + 2\gamma_{k+1}(F(z) - F(y))$$

Nous obtenons alors notre inégalité en prenant le couple optimal $X \sim \mu_{k+1/3}, Z \sim \nu$ tel que $\mathbb{E}[||X - Z||^2] = \mathcal{W}_2^2(\mu_{k+1/3}, \nu)$ □

Nous pouvons maintenant énoncer le théorème suivant :

Théorème 1. *S f est de gradient L-Lipschitz, g est C-Lipschitz et μ_i est la densité de probabilité de Y_i on a :*

$$\begin{aligned} 2\gamma_{k+1}(\mathcal{F}(\mu_{k+1}) - \mathcal{F}(\pi_E)) \leqslant\ & 2\gamma_{k+1}^2(Lp||\,\mathrm{M}^{-1}\,||_2^2 + C^2) \\ & + \mathcal{W}_2^2(\mu_{k+1/3}, \pi_E) \\ & - \mathcal{W}_2^2(\mu_{k+4/3}, \pi_E) \end{aligned} \tag{19}$$

Démonstration. En prenant $\nu = \pi_E$ les lemmes 1 et 2 permettent de contrôler l'incrément de l'énergie potentielle associée à F :

$$\begin{aligned} 2\gamma_{k+1}(\mathcal{E}_F(\mu_{k+2/3}) - \mathcal{E}_F(\pi_E)) =\ & 2Lp||\,\mathrm{M}^{-1}\,||_2^2\gamma_{k+1}^2 \\ & + \mathcal{W}_2^2(\mu_{k+1/3}, \pi_E) \\ & - \mathcal{W}_2^2(\mu_{k+2/3}, \pi_E) \end{aligned}$$

Le lemme 29 de [DMM18] permet de contrôler l'incrément de l'énergie potentielle associée à g :

$$\begin{aligned} 2\gamma_{k+1}(\mathcal{E}_g(\mu_{k+1}) - \mathcal{E}_g(\pi_E)) \leqslant & \mathcal{W}_2^2(\mu_{k+1}, \pi_E) \\ & - \mathcal{W}_2^2(\mu_{k+4/3}, \pi_E) \\ & + 2\gamma_{k+1}^2 C^2 \end{aligned}$$

Finalement le lemme 5 de [DMM18] nous donne une borne sur l'incrément de l'entropie relative

$$\begin{aligned} 2\gamma_{k+1}(\mathcal{H}_E(\mu_{k+1}) - \mathcal{H}(\pi_E)) \leqslant\ & \mathcal{W}_2^2(\mu_{k+2/3}, \pi_E) \\ & - \mathcal{W}_2^2(\mu_{k+1}, \pi_E) \end{aligned}$$

On obtient alors l'inégalité voulue en sommant les trois équations. □

Par convexité de la divergence de Kullback-Leibler nous obtenons un théorème du type théorème 6 de [DMM18] sur la convergence des moyennes des itérés :

Théorème 2. *Posons $\nu_{n+1} = \frac{1}{n+1}\sum_{i=0}^n \mu_i$, $C_3 = 2(Ld||\,\mathrm{M}^{-1}\,||_2^2 + C^2)$ et supposons $\mathcal{W}_2^2(\mu_{1/3}, \pi_E) \leqslant C_4$ où C_4 est une certaine constante alors en prenant un pas constant $\gamma = \frac{\epsilon}{2C_3}$ et $n+1 \geqslant \frac{4C_3C_4}{\epsilon^2}$ nous avons :*

$$\mathrm{KL}(\nu_n, \pi_E) \leqslant \epsilon \tag{20}$$

Démonstration. Par convexité de la KL on a :

$$\begin{aligned} &\mathrm{KL}(\nu_n, \pi_e) \leqslant \frac{1}{n+1}\sum_{k=0}^n \mathrm{KL}(\mu_k, \pi_E) \\ &\leqslant \sum_{k=0}^n \frac{(C_3\gamma + \frac{1}{\gamma}(\mathcal{W}_2^2(\mu_{k+1/3}, \pi_E) - \mathcal{W}_2^2(\mu_{k+4/3}, \pi_E)))}{n+1} \\ &\leqslant C_3\gamma + \frac{1}{\gamma(n+1)}(W_2^2(\mu_{1/3}, \pi_E) - \mathcal{W}_2^2(\mu_{n+4/3}, \pi_E)) \\ &\leqslant C_3\gamma + \frac{C_4}{\gamma(n+1)} \end{aligned} \tag{21}$$

L'inégalité voulue est obtenue en remplaçant γ et n par leurs valeurs respectives. □

Maintenant que la convergence dans l'espace E vers π_E est assurée, on obtient facilement la même borne entre les lois de $X_k = M^{-1}Y_k$ et π notre loi cible.

Théorème 3. *Notons $\tilde{\mu}_i$ la densité de $X_i = \mathrm{M}^{-1}\,Y_i$ et $\tilde{\nu}_n = \frac{1}{n}\sum_{i=1}^n \tilde{\mu}_i$ alors sous les mêmes hypothèse que le théorème 2 nous avons :*

$$\mathrm{KL}(\tilde{\nu}_n, \pi) = \mathrm{KL}(\nu_n, \pi_E) \leqslant \epsilon \tag{22}$$

Démonstration. Cela est simplement dû au fait que KL est invariante par transformation affine. □

5 Expériences numériques

Considérons un graphe non orienté $G = (V, E)$ où V est l'ensemble des nœuds et E l'ensemble des arêtes, une orientation arbitraire de ce graphe est la matrice d'incidence $\nabla : \mathbb{R}^V \to \mathbb{R}^E$ associée à cette orientation. Dans le contexte statistique bayésien décrit dans [WSST16], une réalisation d'un vecteur aléatoire $\theta \in \mathbb{R}^V$ est observée. La loi de ce vecteur aléatoire est paramétrée par un vecteur lui même aléatoire $x \in \mathbb{R}^V$ et de loi *a priori* $p(x) \propto \exp(-\lambda\|\nabla x\|_1)$. La vraisemblance du modèle est définie par $L(x|\theta) \propto \exp(-\|x - \theta\|_2^2)$, et l'estimateur du filtrage de tendance sur le

graphe G est l'estimateur du maximum a posteriori associé à ce modèle bayésien.

Dans cette partie, nous nous proposons d'échantillonner selon la loi *a posteriori* de ce modèle

$$\pi(x|\theta) \propto \exp(-U_\theta(x))$$

où $U_\theta(x) = \frac{1}{2}\|x-\theta\|_2^2 + \lambda\|\nabla x\|_1$. Pour cela, nous commençons par décomposer $\pi(\cdot|\theta)$ comme produit de deux lois indépendantes. Pour cela, notons P la projection orthogonale sur le noyau de la matrice ∇ et Q la projection orthogonale sur l'orthogonal du noyau. Notons également,

$$U_1(x) := \frac{1}{2}\|Q(\theta)-x\|_2^2 + \lambda\|\nabla Q(x)\|_1,$$

et

$$U_2(x) := \frac{1}{2}\|P(\theta)-x\|_2^2.$$

Alors, la relation de Pythagore donne $U_\theta(x) = U_1(Q(x)) + U_2(P(x))$. Pour échantillonner proportionnellement à $\exp(-U_2)$, il suffit d'échantillonner une loi gaussienne réduite centrée en $P(\theta)$ sur le noyau de ∇ (qui est une droite). Pour échantillonner proportionnellement à $\exp(-U_1)$ nous utilisons l'algorithme (PL) sur l'orthogonal du noyau avec $M = \nabla$, $g(y) = \lambda\|y\|_1$ et $f(x) = \frac{1}{2}\|Q(\theta)-x\|^2$. Le calcul de l'opérateur proximal de la post-composition F revient à la résolution d'un système linéaire Laplacien qui peut être réalisée efficacement [Spi10]. En utilisant le graphe "Facebook" de [LK14], nous comparons l'algorithme (PL) à l'algorithme

$$X_{k+1} = \mathrm{prox}_{U_\theta}^{\gamma_{k+1}}(X_k) + \sqrt{2\gamma_{k+1}}W_{k+1}$$

étudié dans [DMM18], que nous appelons "Proximal-Langevin". Le calcul de $\mathrm{prox}_{U_\theta}^{\gamma_{k+1}}$ se ramène à un calcul de l'opérateur proximal de $\|\nabla x\|_1$ qui est connu pour être difficile. Nous utilisons une sous-routine (l'algorithme du gradient projeté pour le problème dual) pour le calculer (voir par exemple [SBHJ16]). Nous représentons $\mathcal{F}(\tilde{\nu}_n)$ (nous estimons cette grandeur en utilisant 100 échantillons) en fonction du temps CPU (en utilisant un coeur d'un CPU de 2800 MHz et 256 GB de RAM).

L'algorithme Proximal-Langevin utilise une sous-routine à chaque étape, ce qui l'empêche de produire des itérés aussi souvent que Passty-Langevin. De plus, Proximal-Langevin est plus lent dans ce cas, ce qui est dû au calcul inexact de $\mathrm{prox}_{U_\theta}^{\gamma_{k+1}}$ à chaque étape.

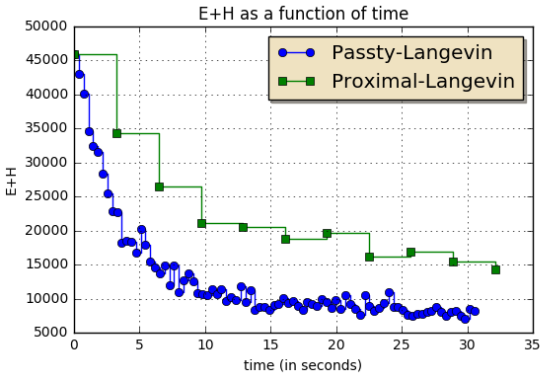


FIGURE 1 – $\mathcal{F}(\tilde{\nu}_n)$ en fonction du temps CPU pour les deux algorithmes

6 Conclusion

Nous présentons dans cet article un nouvel algorithme de type Langevin visant à simuler selon une densité log-concave lorsque la fonction convexe sous-jacente fait intervenir la composée d'une fonction convexe avec un opérateur affine et où l'opérateur proximal de cette composée ne peut pas être calculé facilement. L'exemple typique où ce problème intervient est un cadre bayésien où intervient la norme TV (cf. [DMP18] pour les problèmes d'imagerie computationnelle ou la section 5).

Notre méthode consiste à résoudre le problème dans un espace dual équivalent, mais où il n'est plus nécessaire de calculer l'opérateur proximal correspondant. On montre alors que la vitesse de convergence en nombre d'itérations reste alors la même ($\frac{1}{\epsilon^2}$ itérations pour une précision de ϵ), mais les expériences menées confirment l'avantage de notre méthode en temps CPU.

Ce projet a été soutenu par l'attribution d'une allocation de recherche de la Région Ile-de-France.

Références

[BC11] H. Bauschke and P. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[Ber18] E. Bernton. Langevin Monte Carlo and JKO splitting. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1777–1798. PMLR, 06–09 Jul 2018.

[Dal17] A. Dalalyan. Theoretical guarantees for approximate sampling from a smooth and log-concave density. *J. R. Stat. Soc. B*, 79 :651–676, 2017.

[DMM18] Alain Durmus, Szymon Majewski, and Błażej Miasojedow. Analysis of langevin monte carlo via convex optimization, 2018.

[DMP18] A. Durmus, E. Moulines, and M. Pereyra. Efficient Bayesian Computation by Proximal Markov Chain Monte Carlo : When Langevin Meets Moreau. *SIAM Journal on Imaging Sciences*, 11(1), 2018.

[LK14] Jure Leskovec and Andrej Krevl. SNAP Datasets : Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[NPB14] Neal N. Parikh and S. Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3) :127–239, January 2014.

[Pas79] G. Passty. Ergodic convergence to a zero of the sum of monotone operators in hilbert space. 72 :383–390, 12 1979.

[Per16] M. Pereyra. Proximal markov chain monte carlo algorithms. *Statistics and Computing*, 26(4) :745–760, 2016.

[RT96] G. O. Roberts and R. L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4) :341–363, 12 1996.

[SBHJ16] A. Salim, P. Bianchi, W. Hachem, and J. Jakubowicz. A stochastic proximal point algorithm for total variation regularization over large scale graphs. *IEEE CDC*, 2016.

[Spi10] Daniel A Spielman. Algorithms, graph theory, and linear equations in laplacian matrices. In *Proceedings of the ICM*, volume 4, pages 2698–2722, 2010.

[Vil09] C. Villani. *Optimal Transport : Old and New*. Grundlehren der mathematischen Wissenschaften 338. Springer-Verlag Berlin Heidelberg, 1 edition, 2009.

[Wib18] A. Wibisono. Sampling as optimization in the space of measures : The langevin dynamics as a composite optimization problem. In *COLT*, 2018.

[WSST16] Yu-Xiang Wang, James Sharpnack, Alexander J Smola, and Ryan J Tibshirani. Trend filtering on graphs. *The Journal of Machine Learning Research*, 17(1) :3651–3691, 2016.

Une version corrigée de l'algorithme des plus proches voisins pour l'optimisation de la F-mesure dans un contexte déséquilibré.

Rémi Viola[1,2], Rémi Emonet[1], Amaury Habard[1], Guillaume Metzler[1,3], Sébastien Riou[2], et Marc Sebban[1,2]

[1]Univ. Lyon, UJM Saint-Etienne, CNRS, Institut d'Optique Graduate School, Laboratoire Hubert Curien UMR 5516, F-42023, Saint-Etienne, France
[2]Direction Générances des Finances Publiques, Ministère de l'Economie et des Finances, France
[3]Blitz inc., France

29 mai 2019

Résumé

Dans le présent papier, nous proposons une approche basée sur l'algorithme des plus proches voisins pour de l'apprentissage dans un contexte déséquilibré. Dans un tel contexte, les exemples de la classe minoritaire sont au centre de l'attention et nécessitent des critères d'optimisation spécifiques pour nous permettre de les détecter, comme la F-mesure. Reposant sur des fondements géométriques, nous présentons un algorithme qui pondère la distance entre un nouvel exemple et les exemples positifs de la classe minoritaire. Cela entraîne une modification des régions de Voronoï et donc de la frontière de décision. Une analyse théorique de cette pondération explique comment il est possible de réduire le taux de faux négatifs tout en contrôlant le taux de faux positifs. Les expériences menées sur plusieurs jeux de données publiques, ainsi que sur de grands jeux de données du Ministère de l'Economie et des Finances sur la détection de fraude à l'impôt, mettent en évidence l'efficacité de la méthode en dépit de sa simplicité. En outre, elle se révèle d'autant plus intéressante et performante lorsque qu'elle est combinée à des méthodes d'échantillonage.

Mots-clef : Plus proches voisins, F-mesure, Apprentissage dans un contexte déséquilibré.

1 Introduction

Intrusion detection, health care insurance or bank fraud identification, and more generally anomaly detection, *e.g.* in medicine or in industrial processes, are tasks requiring to address the challenging problem of learning from imbalanced data [Agg17, CBK09]. In such a setting, the training set is composed of a few positive examples (*e.g.* the frauds) and a huge amount of negative samples (*e.g.* the genuine transactions). Standard learning algorithms struggle to deal with this imbalance scenario because they are typically based on the minimization of (a surrogate of) the 0-1 loss. Therefore, a trivial solution consists in assigning the majority label to any test query leading to a high performance from an accuracy perspective but missing the (positive) examples of interest. To overcome this issue, several strategies have been developed over the years. The first one consists in the optimization of loss functions based on measures that are more appropriate for this context such as the *Area Under the ROC Curve* (AUC), the *Average Precision* (AP) or the *F-measure* to cite a few [FHOM09, Ste07]. The main pitfalls related to such a strategy concern the difficulty to directly optimize non smooth, non separable and non convex measures. A simple and usual solution to fix this problem consists in using off-the-shelf learning algorithms (maximizing the accuracy) and a posteriori pick the model with the highest AP or F-Measure. Unfortunately, this might be often suboptimal. A more elaborate solution aims at designing differentiable versions of the previous non-smooth measures and optimizing them, *e.g.* as done by gradient boosting in [FHS$^+$17] with a smooth surrogate of the Mean-AP. The second family of methods is based on the modification of the

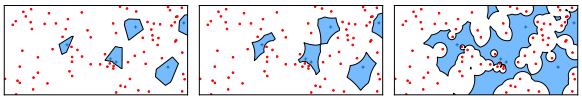


FIGURE 1 – Toy imbalanced dataset : On the left, the Voronoi regions around the positives are small. The risk to generate false negatives (FN) at test time is large. On the right : by increasing too much the regions of influence of the positives, the probability to get false positives (FP) grows. In the middle : an appropriate trade-off between the two previous situations.

distribution of the training data using sampling strategies [FGHC18]. This is typically achieved by removing examples from the majority class, as done, *e.g.*, in *ENN* or *Tomek's Link* [Tom76], and/or by adding examples from the minority class, as in SMOTE [CBHK02] and its variants, or by resorting to generative adversarial models [GPM+14]. One peculiarity of imbalanced datasets can be interpreted from a geometric perspective. As illustrated in Fig. 1 (left) which shows the Voronoi cells on an artificial imbalance dataset (where two adjacent cells have been merged if they concern examples of the same class), the regions of influence of the positive examples are much smaller than that of the negatives. This explains why at test time, in imbalanced learning, the risk to get a false negative is high, leading to a low F-measure, the criterion we focus on in this paper, defined as the harmonic mean of the $Precision = \frac{TP}{TP+FP}$ and the $Recall = \frac{TP}{TP+FN}$, where FP (resp. FN) is the number of false positives (resp. negatives) and TP the number of true positives. Note that increasing the regions of influence of the positives would allow us to reduce FN and improve the F-measure. However, not controlling the expansion of these regions may have a dramatic impact on FP, and so on the F-Measure, as illustrated in Fig. 1 (right).

The main contribution of this paper is about the problem of finding the appropriate trade-off (Fig. 1 (middle)) between the two above-mentioned extreme situations (large FP or FN) both leading to a low F-Measure. A natural way to increase the influence of positives may consist in using generative models (like GANs [GPM+14]) to sample new artificial examples, mimicking the negative training samples. However, beyond the issues related to the parameter tuning, the computation burden and the complexity of such a method, using GANs to optimize the precision and recall is still an open problem (see [SBL+18] for a recent paper on this topic). We show in this paper that a much simpler strategy can be used by modifying the distance exploited in a k-nearest neighbor (NN) algorithm [CH67] which enjoys many interesting advantages, including its simplicity, its capacity to approximate asymptotically any locally regular density, and its theoretical rootedness [LB04, KW15, KSU16]. k-NN also benefited from many algorithmic advances during the past decade in the field of metric learning, aiming at optimizing under constraints the parameters of a metric, typically the Mahalanobis distance, as done in LMNN [WS09] or ITML [DKJ+07] (see [BHS15] for a survey). Unfortunately, existing metric learning methods are dedicated to enhance the k-NN accuracy and do not focus on the optimization of criteria, like the F-measure, in scenarios where the positive training examples are scarce. A geometric solution to increase, at a very low cost, the region of influence of the minority class consists in modifying the distance when comparing a query example to a positive training sample. More formally, we show in this paper that the optimization of the F-Measure is facilitated by weighting the distance to any positive by a coefficient $\gamma \in [0, 1]$ leading to the expansion of the Voronoi cells around the minority examples. An illustration is given in Fig.1 (middle) which might be seen as a good compromise that results in the reduction of FN while controlling the risk to increase FP. Note that our strategy boils down to modifying the local density of the positive examples. For this reason, we claim that it can be efficiently combined with SMOTE-based sampling methods whose goal is complementary and consists in generating examples on the path linking two (potentially far) positive neighbors. Our experiments will confirm this intuition.

The rest of this paper is organized as follows. Section 2 is dedicated to the introduction of our notations. The related work is presented in Section 3. Section 4 is devoted to the presentation of our method. We perform an extensive experimental study in Section 5 on many imbalanced datasets, including non public data from the French Ministry of Economy and Finance on a tax fraud detection task. We give evidence about the complementarity of our method with sampling strategies. We finally conclude in Section 6.

2 Notations and Evaluation Measures

We consider a training sample $S = \{(\mathbf{x}_i, y_i), i = 1, ..., m\}$ of size m, drawn from an unknown joint distribution $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \mathbb{R}^p$ is the feature space and $\mathcal{Y} = \{-1, 1\}$ is the set of labels. Let us as-

sume that $S = S_+ \cup S_-$ with m_+ positives $\in S_+$ and m_- negatives $\in S_-$ where $m = m_+ + m_-$.

Learning from imbalanced datasets requires to optimize appropriate measures that take into account the scarcity of positive examples. Two measures are usually used : the *Recall* or *True Positive Rate* which measures the capacity of the model to recall/detect positive examples, and the *Precision* which is the confidence in the prediction of a positive label :

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{and} \quad \text{Precision} = \frac{TP}{TP + FP},$$

where FP (resp. FN) is the number of false positives (resp. negatives) and TP is the number of true positives. Since one can arbitrarily improve the Precision if there is no constraint on the Recall (and vice-versa), they are usually combined into a single measure : the *F-measure* [Rij79] (or F_1 score), which is widely used in fraud and anomaly detection, and more generally in imbalanced classification [Gee14].

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}}.$$

Note that F_1 considers the Precision and Recall equally.

3 Related Work

In this section, we present the main strategies that have been proposed in the literature to address the problem of learning from imbalanced datasets. We first present methods specifically dedicated to enhance a k-NN classifier. Then, we give an overview of the main sampling strategies used to balance the classes. All these methods will be used in the experimental comparison in Section 5.

3.1 Distance-based Methods

Several strategies have been devised to improve k-NN. The oldest method is certainly the one presented in [Dud76] which consists in associating to each neighbor a voting weight that is inversely proportional to its distance to a query point $\mathbf{x}$. The assigned label $\hat{y}$ of $\mathbf{x}$ is defined as :

$$\hat{y} = \sum_{\mathbf{x}_i \in \text{kNN}(\mathbf{x})} y_i \times \frac{1}{d(\mathbf{x}, \mathbf{x}_i)},$$

where $\text{kNN}(\mathbf{x})$ stands for the set of the k nearest neighbors of $\mathbf{x}$.

In [BSGR03], the authors account both the label and the distance to the neighbors $(\mathbf{x}_i, y_i)$ to define a weighted metric d' from the euclidean distance d, as follows :

$$d'(\mathbf{x}, \mathbf{x}_i) = \left(\frac{m_i}{m}\right)^{1/p} d(\mathbf{x}, \mathbf{x}_i),$$

where m_i is the number of examples in the class y_i. As we will see later, this method falls in the same family of strategies as our contribution, aiming at weighting the distance to the examples according to their label. However, three main differences justify why our method will be better in the experiments : (i) d' is fixed in advance while we will adapt the weight that optimizes the F- measure ; (ii) because of (i), d' needs to take into account the dimension p of the feature space (and so will tend to d as p grows) while this will be intrinsically captured in our method by optimizing the weight given the p-dimensional space ; (iii) d' is useless when combined with sampling strategies (indeed, $\frac{m_i}{m}$ would tend to be uniform) while our method will allow us to weight differently the original positive examples and the ones artificially generated.

Another way to assign weights to each class, which is close to the sampling methods presented in the next section, is to duplicate the positive examples according to the Imbalance Ratio : m_-/m_+. Thus, it can be seen as a *uniform* over-sampling technique, where all positives are replicated the same number of times. However, note that this method requires to work with $k > 1$.

A last family of methods that try to improve k-NN is related to *metric learning*. LMNN [WS09] or ITML [DKJ+07] are two famous examples which optimize under constraints a Mahalanobis distance $d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}_i) = \sqrt{(\mathbf{x} - \mathbf{x}_i)^\top \mathbf{M} (\mathbf{x} - \mathbf{x}_i)}$ parameterized by a positive semidefinite (PSD) matrix $\mathbf{M}$. Such methods seek a linear projection of the data in a latent space where the Euclidean distance is applied. As we will see in the following, our weighting method is a specific case of metric learning which looks for a diagonal matrix - applied only when comparing a query to a positive example - and that behaves well in terms of F-Measure.

3.2 Sampling Strategies

One way to overcome the issues induced by the lack of positive examples is to compensate artificially the imbalance between the two classes. Sampling strategies [FGHC18] have been proven to be very efficient to address this problem. In the following, we overview the most used methods in the literature.

The Synthetic Minority Over-sampling Technique [CBHK02] (SMOTE) over-samples a dataset by creating new synthetic positive data. For each minority example $\mathbf{x}$, it randomly selects one of its k nearest positive neighbors and then creates a new random positive point on the line between this neighbor and $\mathbf{x}$. This is done until some desired ratio is reached.

Borderline-SMOTE [HWM05] is an improvement of the SMOTE algorithm. While the latter generates synthetic points from all positive points, BorderLine-SMOTE only focuses on those having more negatives than positives in their neighborhood. More precisely, new points are generated if the number n of negatives in the k-neighborhood is such that $k/2 \leq n \leq k$.

The Adaptive Synthetic [HBGL08] (ADASYN) sampling approach is also inspired from SMOTE. By using a weighted distribution, it gives more importance to classes that are more difficult to classify, *i.e.* where positives are surrounded by many negatives, and thus generates more synthetic data for these classes.

Two other strategies combine an over-sampling step with an under-sampling procedure. The first one uses the Edited Nearest Neighbors [Wil72] (ENN) algorithm on the top of SMOTE. After SMOTE has generated data, the ENN algorithm removes data that are misclassified by their k nearest neighbors. The second one combines SMOTE with Tomek's link [Tom76]. A Tomek's link is a pair of points $(\mathbf{x}_i, \mathbf{x}_j)$ from different classes for which there is no other point $\mathbf{x}_k$ verifying $d(\mathbf{x}_i, \mathbf{x}_k) \leq d(\mathbf{x}_i, \mathbf{x}_j)$ or $d(\mathbf{x}_k, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_j)$. In other words, $\mathbf{x}_i$ is the nearest neighbor of $\mathbf{x}_j$ and vice-versa. If so, one removes the example of $(\mathbf{x}_i, \mathbf{x}_j)$ that belongs to the majority class. Note both strategies tend to eliminate the overlapping between classes.

Interestingly, we can note that all the previous sampling methods try to overcome the problem of learning from imbalanced data by resorting to the notion of k-neighborhood. This is justified by the fact that k-NN has been shown to be a good estimate of the density at a given point in the feature space. In our contribution, we stay in this line of research. Rather than generating new examples, that would have a negative impact from a complexity perspective, we locally modify the density around the positive points. This is achieved by rescaling the distance between a test sample and the positive training examples. We will show that such a strategy can be efficiently combined with sampling methods, whose goal is complementary, by potentially generating new examples in regions of the space where the minority class is not present.

4 Proposed Approach

In this section, we present our $\gamma k-$NN method which works by scaling the distance between a query point and positive training examples by a factor.

4.1 A Corrected $k-$NN algorithm

Statistically, when learning from imbalanced data, a new query $\mathbf{x}$ has more chance to be close to a negative example due to the rarity of positives in the training set, even around the mode of the positive distribution. We have seen two families of approaches that can be used to counteract this effect : (i) creating new synthetic positive examples, and (ii) changing the distance according to the class. The approach we propose falls into the second category.

We propose to modify how the distance to the positive examples is computed, in order to compensate for the imbalance in the dataset. We artificially bring a new query $\mathbf{x}$ closer to any positive data point $\mathbf{x}_i \in S_+$ in order to increase the effective area of influence of positive examples. The new measure d_γ that we propose is defined, using an underlying distance d (e.g. the euclidean distance) as follows :

$$d_\gamma(\mathbf{x}, \mathbf{x}_i) = \begin{cases} d(\mathbf{x}, \mathbf{x}_i) & \text{if } \mathbf{x}_i \in S_-, \\ \gamma \cdot d(\mathbf{x}, \mathbf{x}_i) & \text{if } \mathbf{x}_i \in S_+. \end{cases}$$

As we will tune the γ parameter, this new way to compute the similarity to a positive example is close to a Mahalanobis-distance learning algorithm, looking for a PSD matrix, as previously described. However, the matrix $\mathbf{M}$ is restricted to be $\gamma^2 \cdot \mathbf{I}$, where $\mathbf{I}$ refers to the identity matrix. Moreover, while metric learning typically works by optimizing a convex loss function under constraints, our γ is simply tuned such as maximizing the non convex F-Measure. Lastly, and most importantly, it is applied only when comparing the query to positive examples. As such, d_γ is not a proper distance, however, it is exactly this which allows it to compensate for the class imbalance. In the binary setting, there is no need to have a γ parameter for the negative class, since only the relative distances are used. In the multi-class setting with K classes, we would have to tune up to $K-1$ values of γ.

Before formalizing the $\gamma k-$NN algorithm that will leverage the distance d_γ, we illustrate in Fig. 2, on 2D data, the decision boundary induced by a nearest neighbor binary classifier that uses d_γ. We consider an elementary dataset with only two points, one positive and one negative. The case of $\gamma = 1$, which is a traditional 1-NN is shown in a thick black line. Lowering

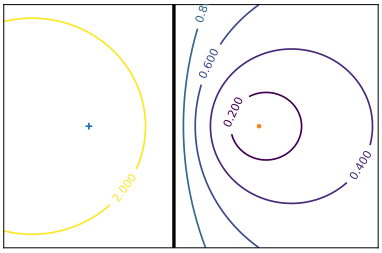


FIGURE 2 – Evolution of the decision boundary based on d_γ, for a 1-NN classifier, on a 2D dataset with one positive (resp. negative) instance represented by a blue cross (resp. orange point). The value of γ is given on each boundary ($\gamma = 1$ on the thick line).

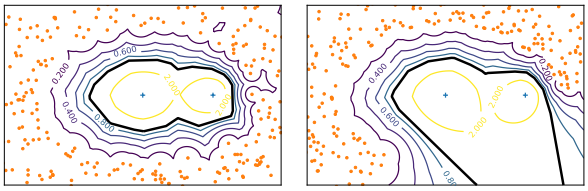


FIGURE 3 – Behavior of the decision boundary according to the γ value for the 1-NN classifier on two toy datasets. The positive points are represented by blue crosses and the negatives by orange points. The black line represents the standard decision boundary for the 1-NN classifier, i.e. when $\gamma = 1$.

the value of γ below 1 brings the decision boundary closer to the negative point, and eventually tends to surround it very closely. In Fig 3, two more complex datasets are shown, each with two positive points and several negative examples. As intuited, we see that the γ parameter allows to control how much we want to push the boundary towards negative examples.

We can now introduce the $\gamma k-$NN algorithm (see Algo 1) that is parameterized by a γ parameter. It has the same overall complexity as $k-$NN. The first step to classify a query $\mathbf{x}$ is to find its k nearest negative neighbors and its k nearest positive neighbors. Then, the distances to the positive neighbors are multiplied by γ, to obtain d_γ. These $2k$ neighbors are then ranked and the k closest ones are used for classification (with a majority vote, as in $k-$NN). It should be noted that, although d_γ does not define a proper distance, we can still use any existing fast nearest neighbor search algorithm, because the actual search is done (twice but) only using the original distance d.

Algorithm 1: Classification of a new example with $\gamma k-$NN

Input : a query $\mathbf{x}$ to be classified, a set of labeled samples $S = S_+ \cup S_-$, a number of neighbors k, a positive real value γ, a distance function d

Output: the predicted label of $\mathbf{x}$

$\mathcal{NN}^-, \mathcal{D}^- \leftarrow nn(k, \mathbf{x}, S_-)$ // nearest negative neighbors with their distances

$\mathcal{NN}^+, \mathcal{D}^+ \leftarrow nn(k, \mathbf{x}, S_+)$ // nearest positive neighbors with their distances

$\mathcal{D}^+ \leftarrow \gamma \cdot \mathcal{D}^+$

$\mathcal{NN}_\gamma \leftarrow$
$firstK\left(k, sortedMerge((\mathcal{NN}^-, \mathcal{D}^-), (\mathcal{NN}^+, \mathcal{D}^+))\right)$

$y \leftarrow +$ if $\left|\mathcal{NN}_\gamma \cap \mathcal{NN}^+\right| \geq \frac{k}{2}$ else $-$ // majority vote based on $\mathcal{NN}_\gamma$

return y

4.2 Theoretical analysis

In this section, we formally analyze what could be a good range of values for the γ parameter of our corrected version of the $k-$NN algorithm. To this aim, we study what impact γ has on the probability to get a false positive (and false negative) at test time and explain why it is important to choose $\gamma < 1$ when the imbalance in the data is significant. The following analysis is made for $k = 1$ but note that the conclusion still holds for a k-NN.

Proposition 1 *(False Negative probability) Let $d_\gamma(\mathbf{x}, \mathbf{x}_+) = \gamma d(\mathbf{x}, \mathbf{x}_+)$, $\forall \gamma > 0$, be our modified distance used between a query $\mathbf{x}$ and any positive training example $\mathbf{x}_+$, where $d(\mathbf{x}, \mathbf{x}_+)$ is some distance function. Let $FN_\gamma(\mathbf{z})$ be the probability for a positive example $\mathbf{z}$ to be a false negative using Algorithm (1). The following result holds : if $\gamma \leq 1$,*

$$FN_\gamma(\mathbf{z}) \quad \leq \quad FN(\mathbf{z})$$

Proof 1 *(sketch of proof) Let ϵ be the distance from $\mathbf{z}$ to its nearest-neighbor $N_\mathbf{z}$. $\mathbf{z}$ is a false negative if $N_\mathbf{z} \in S_-$ that is all positives $\mathbf{x}' \in S_+$ are outside the sphere $\mathcal{S}_{\frac{\epsilon}{\gamma}}(\mathbf{z})$ centered at $\mathbf{z}$ of radius $\frac{\epsilon}{\gamma}$. Therefore,*

$$\begin{aligned} FN_\gamma(\mathbf{z}) &= \prod_{\mathbf{x}' \in S_+} \left(1 - P(\mathbf{x}' \in \mathcal{S}_{\frac{\epsilon}{\gamma}}(\mathbf{z}))\right), \\ &= \left(1 - P(\mathbf{x}' \in \mathcal{S}_{\frac{\epsilon}{\gamma}}(\mathbf{z}))\right)^{m_+} \qquad (1) \end{aligned}$$

while

$$FN(\mathbf{z}) = (1 - P(\mathbf{x}' \in \mathcal{S}_\epsilon(\mathbf{z})))^{m_+}. \qquad (2)$$

Solving (1) $\leq$ (2) implies $\gamma \leq 1$.

This result means that satisfying $\gamma < 1$ allows us to increase the decision boundary around positive examples (as illustrated in Fig. 3), yielding a smaller risk to get false negatives at test time. An interesting comment can be made from Eq.(1) and (2) about their convergence. As m_+ is supposed to be very small in imbalanced datasets, the convergence of $FN(\mathbf{z})$ towards 0 is pretty slow, while one can speed-up this convergence with $FN_\gamma(\mathbf{z})$ by increasing the radius of the sphere $\mathcal{S}_{\frac{\epsilon}{\gamma}}(\mathbf{z})$, that is taking a small value for γ.

Proposition 2 *(False Positive probability) Let $FP_\gamma(\mathbf{z})$ be the probability for a negative example $\mathbf{z}$ to be a false positive using Algorithm (1). The following result holds : if $\gamma \geq 1$,*

$$FP_\gamma(\mathbf{z}) \quad \leq \quad FP(\mathbf{z})$$

Proof 2 *(sketch of proof) Using the same idea as before, we get :*

$$\begin{aligned} FP_\gamma(\mathbf{z}) &= \prod_{\mathbf{x}' \in S_-} \left(1 - P(\mathbf{x}' \in \mathcal{S}_{\gamma\epsilon}(\mathbf{z}))\right), \\ &= \left(1 - P(\mathbf{x}' \in \mathcal{S}_{\gamma\epsilon}(\mathbf{z}))\right)^{m_-} \end{aligned} \tag{3}$$

while

$$FP(\mathbf{z}) = \left(1 - P(\mathbf{x}' \in \mathcal{S}_{\epsilon}(\mathbf{z}))\right)^{m_-}. \tag{4}$$

Solving (3) $\leq$ (4) implies $\gamma \geq 1$.

As expected, this result suggests to take $\gamma > 1$ to increase the distance $d_\gamma(\mathbf{z}, \mathbf{x}_+)$ from a negative test sample $\mathbf{z}$ to any positive training example $\mathbf{x}_+$ and thus reduce the risk to get a false positive. It is worth noticing that while the two conclusions from Propositions 1 and 2 are contradictory, the convergence of $FP_\gamma(\mathbf{z})$ towards 0 is much faster than that of $FN_\gamma(\mathbf{z})$ because $m_- >> m_+$ in an imbalance scenario. Therefore, fulfilling the requirement $\gamma > 1$ is much less important than satisfying $\gamma < 1$. For this reason, we will impose our Algorithm (1) to take $\gamma \in]0, 1[$. As we will see in the experimental section, the more imbalance the datasets, the smaller the optimal γ, confirming the previous conclusion.

5 Experiments

In this section, we present an experimental evaluation of our method on public and real private datasets with comparisons to classic distance-based methods and state of the art sampling strategies able to deal with imbalanced data. All results are reported using $k = 3$. The experiments with $k = 1$, fully reported in the supplementary material, follow the same trends.

5.1 Experimental setup

For the experiments, we use several public datasets from the classic UCI [1] and KEEL [2] repositories. We also use eleven real fraud detection datasets provided by the General Directorate of Public Finances (DGFiP) which is part of the French central public administration related to the French Ministry for the Economy and Finance. These private datasets correspond to data coming from tax and VAT declarations of French companies and are used for tax fraud detection purpose covering declaration of over-valued, fictitious or prohibited charges, wrong turnover reduction or particular international VAT frauds such as "VAT carousels". The DGFiP performs about 50,000 tax audits per year within a panel covering more than 3,000,000 companies. Being able to select the right companies to control each year is a crucial issue with a potential high societal impact. Thus, designing efficient imbalance learning methods is key. The main properties of the datasets are summarized in Table 5.1, including the imbalance ratio (IR).

All the datasets are normalized using a min-max normalization such that each feature lies in the range $[-1, 1]$. We randomly draw 80%-20% splits of the data to generate the training and test sets respectively. Hyperparameters are tuned with a 10-fold cross-validation over the training set. We repeat the process over 5 runs and average the results in terms of F-measure F_1. In a first series of experiments, we compare our method, named $\gamma k-$NN, to 6 other distance-based baselines :

- the classic $k-$Nearest Neighbor algorithm ($k-$NN),
- the weighted version of $k-$NN using the inverse distance as a weight to predict the label (w$k-$NN) [Dud76],
- the class weighted version of $k-$NN (cw$k-$NN) [BSGR03],
- the $k-$NN version where each positive is duplicated according to the IR of the dataset (dup$k-$NN),
- the metric learning method LMNN [WS09].

1. https://archive.ics.uci.edu/ml/datasets.html
2. https://sci2s.ugr.es/keel/datasets.php

DATASETS	SIZE	DIM	%+	%−	IR
BALANCE	625	4	46.1	53.9	1.2
AUTOMPG	392	7	37.5	62.5	1.7
IONOSPHERE	351	34	35.9	64.1	1.8
PIMA	768	8	34.9	65.1	1.9
WINE	178	13	33.1	66.9	2
GLASS	214	9	32.7	67.3	2.1
GERMAN	1000	23	30	70	2.3
VEHICLE	846	18	23.5	76.5	3.3
HAYES	132	4	22.7	77.3	3.4
SEGMENTATION	2310	19	14.3	85.7	6
ABALONE8	4177	10	13.6	86.4	6.4
YEAST3	1484	8	11	89	8.1
PAGEBLOCKS	5473	10	10.2	89.8	8.8
SATIMAGE	6435	36	9.7	90.3	9.3
LIBRAS	360	90	6.7	93.3	14
WINE4	1599	11	3.3	96.7	29.2
YEAST6	1484	8	2.4	97.6	41.4
ABALONE17	4177	10	1.4	98.6	71.0
ABALONE20	4177	10	0.6	99.4	159.7
DGFIP 19 2	16643	265	35.1	64.9	1.9
DGFIP 9 2	440	173	24.8	75.2	3
DGFIP 4 2	255	82	20.8	79.2	3.8
DGFIP 8 1	1028	255	17.8	82.2	4.6
DGFIP 8 2	1031	254	17.9	82.1	4.6
DGFIP 9 1	409	171	16.4	83.6	5.1
DGFIP 4 1	240	76	16.2	83.8	5.2
DGFIP 16 1	789	162	10.3	89.7	8.7
DGFIP 16 2	786	164	9.9	90.1	9.1
DGFIP 20 3	17584	294	5	95	19
DGFIP 5 3	19067	318	3.9	96.1	24.9

TABLE 1 – Information about the studied datasets sorted by imbalance ratio. The first part refers to the public datasets, the second one describes the *DGFiP* private datasets.

We set the number of nearest neighbors to $k = 3$ for all methods. The hyperparameter μ of *LMNN*, weighting the impact of impostor constraints (see [WS09] for more details), is tuned in the range $[0, 1]$ using a step of 0.1. Our γ parameter is tuned in the range $[0, 1]$ [3] using a step of 0.1.

In a second series of experiments, we compare our method to the five oversampling strategies described in Section 3.2 : SMOTE, Borderline-SMOTE, ADASYN, SMOTE with ENN, SMOTE with Tomek's link. The number of generated positive examples is tuned over the set of ratios $\frac{m_+}{m_-} \in \{0.1, 0.2, ..., 0.9, 1.0\}$ and such that the new ratio is greater than the original one before sampling. Other parameters of these methods are the default ones used by the package *ImbalancedLearn* of *Scikit-learn*.

3. We experimentally noticed that using a larger range for γ leads in fact to a potential decrease of performances due to overfitting phenomena. This behavior is actually in line with the analysis provided in Section 4.2.

5.2 Results

The results on the public datasets using distance-based methods are provided in Table 5.2. Overall, our $\gamma k-$NN approach performs much better than its competitors by achieving an improvement of at least 3 points on average, compared to the 2nd best method (DUP$k-$NN). The different $k-$NN versions fail globally to provide models efficient whatever the imbalance ratio. The metric learning approach LMNN is competitive when IR is smaller than 10 (although algorithmically more costly). Beyond, it faces some difficulties to find a relevant projection space due to the lack of positive data. The efficiency of $\gamma k-$NN is not particularly sensitive to the imbalance ratio.

The results for our second series of experiments, focusing on sampling strategies, are reported on Fig. 4. We compare each of the 5 sampling methods with the average performances of 3−NN and γk-NN obtained over the 19 public datasets reported in Table 5.2. Additionally, we also use $\gamma k-$NN on the top of the sampling methods to evaluate how both strategies are complementary. However, in this scenario, we propose to learn a different γ value to be used with the synthetic positives. Indeed, some of them may be generated in some true negative areas and in this situation it might be more appropriate to decrease the influence of such synthetic examples. The γ parameter for these examples is then tuned in the range $[0, 2]$ using a step of 0.1. If one can easily observe that all the oversampling strategies improve the classic $k-NN$, none of them is better than our γk-NN method showing that our approach is able to deal efficiently with imbalanced data. Moreover, we are able to improve the efficiency of γk-NN when it is coupled with an oversampling strategy. The choice of the oversampler does not really influence the results. The gains obtained by using a sampling method with γk-NN for each dataset is illustrated in Fig. 6 (left).

To study the influence of using two γ parameters when combined with an oversampling strategy, we show an illustration (Fig. 5 (top)) of the evolution of the F-measure with respect to the γ values for synthetic and real positive instances. The best F-measure is achieved when the γ on real positives is smaller than 1 and when the γ on synthetic positives is greater than 1, justifying the interest of using two parameterizations of γ. In Fig. 5 (bottom), we show how having two γ values gives the flexibility to independently control the increased influence of real positives and the one of artificial

DATASETS	3−NN	DUPk−NN	Wk−NN	CWk−NN	LMNN	γk−NN
BALANCE	0.954(0.017)	0.954(0.017)	0.957(0.017)	0.961(0.010)	**0.963**(0.012)	0.954(0.029)
AUTOMPG	0.808(0.077)	0.826(0.033)	0.810(0.076)	0.815(0.053)	0.827(0.054)	**0.831**(0.025)
IONOSPHERE	0.752(0.053)	0.859(0.021)	0.756(0.060)	0.799(0.036)	0.890(0.039)	**0.925**(0.017)
PIMA	0.500(0.056)	0.539(0.033)	0.479(0.044)	0.515(0.037)	0.499(0.070)	**0.560**(0.024)
WINE	0.881(0.072)	0.852(0.057)	0.881(0.072)	0.876(0.080)	**0.950**(0.036)	0.856(0.086)
GLASS	0.727(0.049)	0.733(0.061)	0.736(0.052)	0.717(0.055)	0.725(0.048)	**0.746**(0.046)
GERMAN	0.330(0.030)	0.449(0.037)	0.326(0.030)	0.344(0.029)	0.323(0.054)	**0.464**(0.029)
VEHICLE	0.891(0.044)	0.867(0.027)	0.891(0.044)	0.881(0.021)	**0.958**(0.020)	0.880(0.049)
HAYES	0.036(0.081)	0.183(0.130)	0.050(0.112)	0.221(0.133)	0.036(0.081)	**0.593**(0.072)
SEGMENTATION	0.859(0.028)	0.862(0.018)	0.877(0.028)	0.851(0.022)	**0.885**(0.034)	0.848(0.025)
ABALONE8	0.243(0.037)	0.318(0.013)	0.241(0.034)	0.330(0.015)	0.246(0.065)	**0.349**(0.018)
YEAST3	0.634(0.066)	0.670(0.034)	0.634(0.066)	**0.699**(0.015)	0.667(0.055)	0.687(0.033)
PAGEBLOCKS	0.842(0.020)	0.850(0.024)	0.849(0.019)	0.847(0.029)	**0.856**(0.032)	0.844(0.023)
SATIMAGE	0.454(0.039)	0.457(0.027)	0.454(0.039)	0.457(0.023)	**0.487**(0.026)	0.430(0.008)
LIBRAS	**0.806**(0.076)	0.788(0.187)	**0.806**(0.076)	0.789(0.097)	0.770(0.027)	0.768(0.106)
WINE4	0.031(0.069)	**0.090**(0.086)	0.031(0.069)	0.019(0.042)	0.000(0.000)	**0.090**(0.036)
YEAST6	0.503(0.302)	0.449(0.112)	0.502(0.297)	0.338(0.071)	0.505(0.231)	**0.553**(0.215)
ABALONE17	0.057(0.078)	**0.172**(0.086)	0.057(0.078)	0.096(0.059)	0.000(0.000)	0.100(0.038)
ABALONE20	0.000(0.000)	0.000(0.000)	0.000(0.000)	**0.067**(0.038)	0.057(0.128)	0.052(0.047)
MEAN	0.543(0.063)	0.575(0.053)	0.544(0.064)	0.559(0.046)	0.560(0.053)	**0.607**(0.049)

TABLE 2 – Results for 3−NN on the public datasets. The values correspond to the mean F-measure F_1 over 5 runs. The standard deviation is indicated between brackets. The best result on each dataset is indicated in bold.

positives.

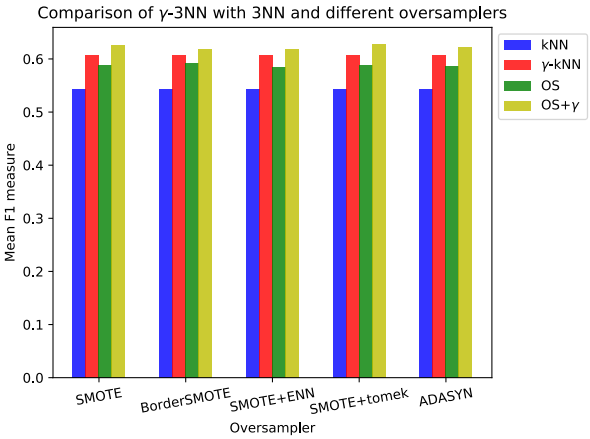


FIGURE 4 – Comparison of different sampling strategies averaged over the 19 public datasets. OS refers to the results of the corresponding sampling strategy and $OS+\gamma$ to the case when the sampling strategy is combined with γk-NN. k−NN and γk−NN refers to the results of these methods without oversampling as obtained in Table 5.2. (numerical values for these graphs are provided in supplementary material)

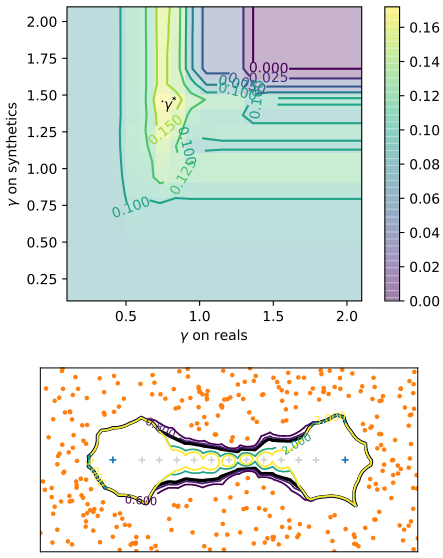


FIGURE 5 – (Top) An example of heatmap that shows the best couple of γ for the OS+γk−NN strategy on the yeast6 dataset with SMOTE and Tomek's link. (Bottom) Illustration, on a toy dataset, of the effect of varying the γ for generated positive points (in grey) while keeping a fixed $\gamma = 0.4$ for real positive points.

We now propose a study on the influence of the imbalance ratio on the optimal γ-parameter. We consider the *Balance* dataset which has the smallest imbalance ratio that we increase by iteratively randomly under-

sampling the minority class over the training set. We report the results on Fig. 6 (right). As expected, we can observe that the optimal γ value decreases when the imbalance increases. However, note that from a certain IR (around 15), γ stops decreasing to be able to keep a satisfactory F-Measure.

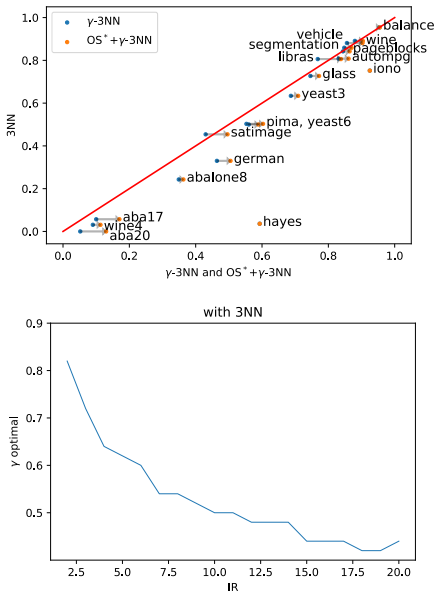


FIGURE 6 – (Top) Comparison of k-NN with (i) $\gamma k-$NN (points in blue) and (ii) $\gamma k-$NN coupled with the best sampling strategy (OS*) (points in orange) for each dataset and for $k = 3$. Points below the line $y = x$ means that k-NN is outperformed. (Bottom) Evolution of the optimal γ value with respect to the IR for $k = 3$.

The results for the real datasets of the DGFiP are available in Table 3. Note that only the SMOTE algorithm is reported here since the other oversamplers have comparable performances. The analysis of the results leads to observations similar as the ones made for the public datasets. Our $\gamma - k$NN approach outperforms classic $k-$NN and is better than the results obtained by the SMOTE strategy. Coupling the SMOTE sampling method with our distance correction γk-NN allows us to improve the global performance showing the applicability of our method on real data.

6 Conclusion

In this paper, we have proposed a new strategy that addresses the problem of learning from imbalanced datasets, based on the $k-$NN algorithm and that modifies the distance to the positive examples. It has been shown to outperform its competitors in term of F_1-measure. Furthermore, the proposed approach is complementary to oversampling strategies and can even increase their performance. Our $\gamma k-$NN algorithm, despite its simplicity, is highly effective even on real data sets.

Two lines of research deserve future investigations. We can note that tuning γ is equivalent to building a diagonal matrix (with γ^2 in the diagonal) and applying a Mahalanobis distance only between a query and a positive example. This comment opens the door to a new metric learning algorithm dedicated to optimizing a PSD matrix under F-Measure-based constraints. If one can learn such a matrix, the second perspective will consist in deriving generalization guarantees over the learned matrix. In addition, making γ non-stationary (a $\gamma(\mathbf{x})$ that smoothly varies in $\mathcal{X}$) would increase the model flexibility.

Références

[Agg17] Charu C. Aggarwal. *Outlier Analysis*. Springer International Publishing, 2017.

[BHS15] Aurélien Bellet, Amaury Habrard, and Marc Sebban. *Metric Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2015.

[BSGR03] Ricardo Barandela, José Salvador Sánchez, V Garca, and Edgar Rangel. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36, 2003.

[CBHK02] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote : synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 2002.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection : A survey. *ACM Comput. Surv.*, 2009.

[CH67] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13, 1967.

[DKJ+07] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *ICML*, 2007.

DATASETS	3−NN	γk−NN	SMOTE	SMOTE+γk−NN
DGFIP19 2	0,454$_{(0,007)}$	0,528$_{(0,005)}$	0,505$_{(0,010)}$	**0,529**$_{(0,003)}$
DGFIP9 2	0,173$_{(0,074)}$	0,396$_{(0,018)}$	0,340$_{(0,033)}$	**0,419**$_{(0,029)}$
DGFIP4 2	0,164$_{(0,155)}$	0,373$_{(0,018)}$	0,368$_{(0,057)}$	**0,377**$_{(0,018)}$
DGFIP8 1	0,100$_{(0,045)}$	**0,299**$_{(0,010)}$	0,278$_{(0,043)}$	**0,299**$_{(0,011)}$
DGFIP8 2	0,140$_{(0,078)}$	0,292$_{(0,028)}$	**0,313**$_{(0,048)}$	0,312$_{(0,021)}$
DGFIP9 1	0,088$_{(0,090)}$	0,258$_{(0,036)}$	0,270$_{(0,079)}$	**0,288**$_{(0,026)}$
DGFIP4 1	0,073$_{(0,101)}$	0,231$_{(0,139)}$	0,199$_{(0,129)}$	**0,278**$_{(0,067)}$
DGFIP16 1	0,049$_{(0,074)}$	0,166$_{(0,065)}$	0,180$_{(0,061)}$	**0,191**$_{(0,081)}$
DGFIP16 2	0,210$_{(0,102)}$	0,202$_{(0,056)}$	0,220$_{(0,043)}$	**0,229**$_{(0,026)}$
DGFIP20 3	0,142$_{(0,015)}$	0,210$_{(0,019)}$	0,199$_{(0,015)}$	**0,212**$_{(0,019)}$
DGFIP5 3	0,030$_{(0,012)}$	0,105$_{(0,008)}$	**0,110**$_{(0,109)}$	0,107$_{(0,010)}$
MEAN	0,148$_{(0,068)}$	0,278$_{(0,037)}$	0,271$_{(0,057)}$	**0,295**$_{(0,028)}$

TABLE 3 – Results for 3−NN on the DGFiP datasets. The values correspond to the mean F-measure F_1 over 5 runs. The best result on each dataset is indicated in bold while the second is underlined.

[Dud76] Sahibsingh A Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 4, 1976.

[FGHC18] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. Smote for learning from imbalanced data : Progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, 61, 2018.

[FHOM09] César Ferri, José Hernández-Orallo, and R Modroiu. An experimental comparison of performance measures for classification. *PRL*, 30, 2009.

[FHS+17] Jordan Fréry, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Efficient top rank optimization with gradient boosting for supervised anomaly detection. In *ECML-PKDD*, 2017.

[Gee14] Sunder Gee. *Fraud and fraud detection : a data analytics approach.* 2014.

[GPM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[HBGL08] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn : Adaptive synthetic sampling approach for imbalanced learning. In *IJCNN*, 2008.

[HWM05] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote : a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, 2005.

[KSU16] Aryeh Kontorovich, Sivan Sabato, and Ruth Urner. Active nearest-neighbor learning in metric spaces. In *NIPS*. 2016.

[KW15] Aryeh Kontorovich and Roi Weiss. A Bayes consistent 1-NN classifier. In *ICAIS*, volume 38, 2015.

[LB04] Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with lipschitz functions. *JMLR*, 5, 2004.

[Rij79] C. J. Van Rijsbergen. *Information Retrieval.* 1979.

[SBL+18] Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *NeurIPS*. 2018.

[Ste07] Harald Steck. Hinge rank loss and the area under the roc curve. In Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *Machine Learning : ECML 2007*, 2007.

[Tom76] Ivan Tomek. Two modifications of cnn. In *IEEE Transactions on Systems Man and Communications*, 1976.

[Wil72] Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, 3, 1972.

[WS09] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10, 2009.

Annexe : Une version corrigée de l'algorithme des plus proches voisins pour l'optimisation de la F-mesure dans un contexte déséquilibré.

Rémi Viola[1,2], Rémi Emonet[1], Amaury Habard[1], Guillaume Metzler[1,3], Sébastien Riou[2], et Marc Sebban[1,2]

[1]Univ. Lyon, UJM Saint-Etienne, CNRS, Institut d'Optique Graduate School, Laboratoire Hubert Curien UMR 5516, F-42023, Saint-Etienne, France
[2]Direction Générances des Finances Publiques, Ministère de l'Economie et des Finances, France
[3]Blitz inc., France

29 mai 2019

In the main paper, all the results were presented using a 3−NN algorithm. The given Supplementary Materiel provides all the tables used to generate the different graphs presented in the main paper.
We also provide graphs and tables that illustrate our approach using a Nearest Neighbor Algorithm, i.e. a 1−NN algorithm.

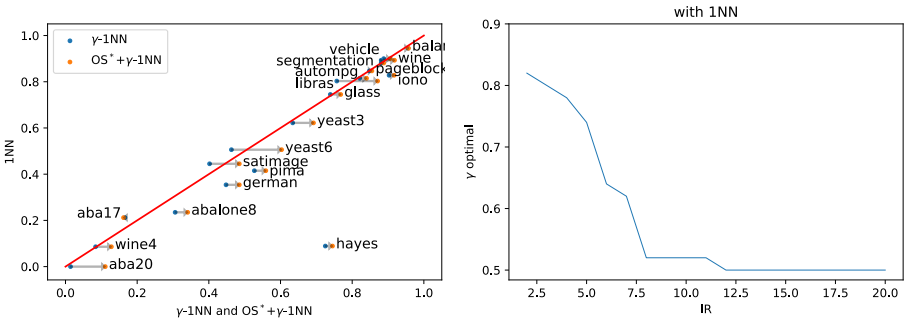


FIGURE 1 – On the left : Comparison of k-NN with (i) $\gamma k-$NN (points in blue) and (ii) $\gamma k-$NN coupled with the best sampling strategy (OS*) (points in orange) for each dataset and for $k = 1$. Points below the line $y = x$ means that k-NN is outperformed. On the right : Evolution of the optimal γ value with respect to the IR for $k = 1$.

TABLE 1 – Results for 1−NN on the public datasets. The values correspond to the mean F-measure F_1 over 5 runs. The best result on each dataset is indicated in bold.

DATASETS	1−NN	DUPk−NN	Wk−NN	CWk−NN	LMNN	γk−NN
BALANCE	0.945(0.016)	0.945(0.016)	0.945(0.016)	0.943(0.023)	**0.972**(0.017)	0.956(0.022)
AUTOMPG	0.815(0.045)	0.815(0.045)	0.815(0.045)	**0.826**(0.045)	0.813(0.040)	0.821(0.044)
IONOSPHERE	0.828(0.039)	0.828(0.039)	0.828(0.039)	0.842(0.046)	0.862(0.034)	**0.903**(0.031)
PIMA	0.415(0.040)	0.415(0.040)	0.415(0.040)	0.463(0.048)	0.429(0.032)	**0.527**(0.032)
WINE	0.893(0.055)	0.893(0.055)	0.893(0.055)	0.880(0.072)	**0.918**(0.064)	0.881(0.069)
GLASS	0.745(0.105)	0.745(0.105)	0.745(0.105)	**0.780**(0.082)	0.742(0.084)	0.739(0.065)
GERMAN	0.354(0.059)	0.354(0.059)	0.354(0.059)	0.387(0.046)	0.343(0.050)	**0.448**(0.020)
VEHICLE	0.900(0.045)	0.900(0.045)	0.900(0.045)	0.891(0.045)	**0.951**(0.009)	0.888(0.043)
HAYES	0.089(0.199)	0.133(0.199)	0.089(0.199)	0.263(0.163)	0.146(0.208)	**0.725**(0.165)
SEGMENTATION	0.881(0.035)	0.881(0.035)	0.881(0.035)	0.876(0.032)	**0.911**(0.024)	0.881(0.035)
ABALONE8	0.235(0.018)	0.235(0.018)	0.235(0.018)	0.298(0.019)	0.237(0.017)	**0.306**(0.006)
YEAST3	0.622(0.045)	0.622(0.045)	0.622(0.045)	0.613(0.035)	0.613(0.050)	**0.634**(0.044)
PAGEBLOCKS	0.847(0.022)	**0.848**(0.025)	0.847(0.022)	0.846(0.021)	0.843(0.017)	**0.848**(0.023)
SATIMAGE	0.445(0.015)	0.445(0.015)	0.445(0.015)	0.442(0.012)	**0.475**(0.025)	0.402(0.015)
LIBRAS	0.803(0.129)	0.803(0.129)	0.803(0.129)	**0.826**(0.154)	0.803(0.129)	0.757(0.055)
WINE4	0.086(0.085)	0.086(0.085)	0.086(0.085)	**0.113**(0.065)	0.041(0.057)	0.084(0.052)
YEAST6	**0.506**(0.155)	**0.506**(0.155)	**0.506**(0.155)	0.271(0.049)	0.485(0.144)	0.463(0.106)
ABALONE17	**0.212**(0.084)	**0.212**(0.084)	**0.212**(0.084)	0.095(0.038)	0.128(0.062)	0.166(0.052)
ABALONE20	0.000(0.000)	0.000(0.000)	0.000(0.000)	**0.036**(0.036)	0.000(0.000)	0.014(0.032)
MEAN	0.559(0.063)	0.561(0.063)	0.559(0.063)	0.563(0.054)	0.564(0.056)	**0.602**(0.048)

TABLE 2 – Results for 1−NN on the public datasets. The values correspond to the mean F-measure F_1 over 5 runs. The best result on each dataset is indicated in bold.

DATASETS	SMOTE	SMOTE γk−NN	BORDER SMOTE	B-SMOTE γk−NN	SMOTE +ENN	SM+ENN γk−NN	SMOTE +TOMEK	SM+TOMEK γk−NN	ADASYN	ADASYN $+\gamma k$−NN
BALANCE	0.947	**0.957**	0.949	0.947	0.917	0.946	0.942	**0.957**	0.948	0.951
AUTOMPG	0.818	0.833	0.825	0.826	**0.843**	0.840	0.829	0.834	0.821	0.828
IONOSPHERE	0.862	**0.917**	0.858	0.912	0.859	0.909	0.862	**0.917**	0.863	0.908
PIMA	0.431	0.531	0.431	0.528	0.555	**0.559**	0.469	0.544	0.441	0.540
WINE	0.906	0.897	0.903	0.906	0.870	**0.918**	0.906	0.908	0.907	0.913
GLASS	0.748	0.768	**0.782**	0.746	0.729	0.759	0.750	0.750	**0.782**	0.753
GERMAN	0.397	0.462	0.394	0.454	0.479	**0.485**	0.406	0.464	0.395	0.454
VEHICLE	0.896	0.899	0.905	0.901	0.854	0.875	0.896	**0.906**	0.904	0.899
HAYES	0.356	**0.745**	0.273	**0.745**	0.307	0.430	0.394	**0.745**	0.307	**0.745**
SEGMENTATION	0.883	0.883	0.883	0.881	0.822	0.843	**0.888**	**0.888**	0.878	0.884
ABALONE8	0.277	0.320	0.293	0.335	0.338	**0.341**	0.281	0.320	0.281	0.316
YEAST3	0.637	0.643	0.632	0.628	0.682	**0.692**	0.653	0.648	0.639	0.627
PAGEBLOCKS	0.848	0.852	0.841	0.850	0.840	0.852	0.852	**0.855**	0.840	0.849
SATIMAGE	0.457	0.457	0.462	0.464	0.439	**0.485**	0.457	0.453	0.463	0.464
LIBRAS	0.845	**0.871**	0.865	0.856	0.723	0.721	0.845	**0.871**	0.865	0.869
WINE4	0.123	0.091	0.119	0.128	0.097	0.082	**0.130**	0.086	0.103	0.088
YEAST6	0.495	0.537	0.495	0.552	0.399	**0.603**	0.528	0.522	0.449	0.533
ABALONE17	0.139	0.161	**0.164**	0.159	0.127	0.151	0.139	0.162	0.142	0.160
ABALONE20	**0.116**	0.109	0.080	0.060	0.082	0.111	**0.116**	0.107	0.115	0.108
MEAN	0.588	**0.628**	0.587	0.625	0.577	0.611	0.597	**0.628**	0.586	0.626

TABLE 3 – Results for 1−NN on the DGFIP datasets. The values correspond to the mean F-measure F_1 over 5 runs. The best result on each dataset is indicated in bold while the second is underlined.

DATASETS	1−NN	γk−NN	SMOTE	SMOTE+γk−NN
DGFIP19 2	$0.448_{(0.008)}$	$\underline{0.521}_{(0.004)}$	$0.470_{(0.011)}$	$\mathbf{0.522}_{(0.005)}$
DGFIP9 2	$0.250_{(0.053)}$	$\underline{0.390}_{(0.018)}$	$0.330_{(0.049)}$	$\mathbf{0.404}_{(0.031)}$
DGFIP4 2	$0.205_{(0.071)}$	$\underline{0.349}_{(0.027)}$	$0.270_{(0.071)}$	$\mathbf{0.362}_{(0.036)}$
DGFIP8 1	$0.162_{(0.077)}$	$\mathbf{0.303}_{(0.004)}$	$0.245_{(0.062)}$	$\mathbf{0.303}_{(0.003)}$
DGFIP8 2	$0.210_{(0.022)}$	$\underline{0.309}_{(0.010)}$	$0.282_{(0.073)}$	$\mathbf{0.317}_{(0.014)}$
DGFIP9 1	$0.249_{(0.071)}$	$\underline{0.274}_{(0.025)}$	$0.272_{(0.062)}$	$\mathbf{0.299}_{(0.030)}$
DGFIP4 1	$0.060_{(0.083)}$	$\mathbf{0.261}_{(0.082)}$	$0.174_{(0.124)}$	$\underline{0.246}_{(0.074)}$
DGFIP16 1	$0.083_{(0.063)}$	$\underline{0.183}_{(0.017)}$	$0.144_{(0.032)}$	$\mathbf{0.195}_{(0.033)}$
DGFIP16 2	$0.158_{(0.063)}$	$0.158_{(0.063)}$	$\underline{0.170}_{(0.056)}$	$\mathbf{0.178}_{(0.048)}$
DGFIP20 3	$0.163_{(0.029)}$	$\underline{0.187}_{(0.011)}$	$\mathbf{0.190}_{(0.004)}$	$0.186_{(0.005)}$
DGFIP5 3	$0.081_{(0.011)}$	$0.096_{(0.006)}$	$\mathbf{0.115}_{(0.011)}$	$\underline{0.110}_{(0.007)}$
MEAN	$0.188_{(0.050)}$	$\underline{0.276}_{(0.024)}$	$0.242_{(0.050)}$	$\mathbf{0.284}_{(0.026)}$

TABLE 4 – Results for 3−NN on the public datasets. The values correspond to the mean F-measure F_1 over 5 runs. The best result on each dataset is indicated in bold.

DATASETS	3−NN	DUPk−NN	Wk−NN	CWk−NN	LMNN	γk−NN
BALANCE	0.954(0.017)	0.954(0.017)	0.957(0.017)	0.961(0.010)	**0.963**(0.012)	0.954(0.029)
AUTOMPG	0.808(0.077)	0.826(0.033)	0.810(0.076)	0.815(0.053)	0.827(0.054)	**0.831**(0.025)
IONOSPHERE	0.752(0.053)	0.859(0.021)	0.756(0.060)	0.799(0.036)	0.890(0.039)	**0.925**(0.017)
PIMA	0.500(0.056)	0.539(0.033)	0.479(0.044)	0.515(0.037)	0.499(0.070)	**0.560**(0.024)
WINE	0.881(0.072)	0.852(0.057)	0.881(0.072)	0.876(0.080)	**0.950**(0.036)	0.856(0.086)
GLASS	0.727(0.049)	0.733(0.061)	0.736(0.052)	0.717(0.055)	0.725(0.048)	**0.746**(0.046)
GERMAN	0.330(0.030)	0.449(0.037)	0.326(0.030)	0.344(0.029)	0.323(0.054)	**0.464**(0.029)
VEHICLE	0.891(0.044)	0.867(0.027)	0.891(0.044)	0.881(0.021)	**0.958**(0.020)	0.880(0.049)
HAYES	0.036(0.081)	0.183(0.130)	0.050(0.112)	0.221(0.133)	0.036(0.081)	**0.593**(0.072)
SEGMENTATION	0.859(0.028)	0.862(0.018)	0.877(0.028)	0.851(0.022)	**0.885**(0.034)	0.848(0.025)
ABALONE8	0.243(0.037)	0.318(0.013)	0.241(0.034)	0.330(0.015)	0.246(0.065)	**0.349**(0.018)
YEAST3	0.634(0.066)	0.670(0.034)	0.634(0.066)	**0.699**(0.015)	0.667(0.055)	0.687(0.033)
PAGEBLOCKS	0.842(0.020)	0.850(0.024)	0.849(0.019)	0.847(0.029)	**0.856**(0.032)	0.844(0.023)
SATIMAGE	0.454(0.039)	0.457(0.027)	0.454(0.039)	0.457(0.023)	**0.487**(0.026)	0.430(0.008)
LIBRAS	**0.806**(0.076)	0.788(0.187)	**0.806**(0.076)	0.789(0.097)	0.770(0.027)	0.768(0.106)
WINE4	0.031(0.069)	**0.090**(0.086)	0.031(0.069)	0.019(0.042)	0.000(0.000)	**0.090**(0.036)
YEAST6	0.503(0.302)	0.449(0.112)	0.502(0.297)	0.338(0.071)	0.505(0.231)	**0.553**(0.215)
ABALONE17	0.057(0.078)	**0.172**(0.086)	0.057(0.078)	0.096(0.059)	0.000(0.000)	0.100(0.038)
ABALONE20	0.000(0.000)	0.000(0.000)	0.000(0.000)	**0.067**(0.038)	0.057(0.128)	0.052(0.047)
MEAN	0.543(0.063)	0.575(0.053)	0.544(0.064)	0.559(0.046)	0.560(0.053)	**0.607**(0.049)

TABLE 5 – Results for 3−NN on the public datasets. The values correspond to the mean F-measure F_1 over 5 runs. The best result on each dataset is indicated in bold.

DATASETS	SMOTE	SMOTE γk−NN	BORDER SMOTE	B-SMOTE γk−NN	SMOTE +ENN	SM+ENN γk−NN	SMOTE +TOMEK	SM+TOMEK γk−NN	ADASYN	ADASYN $+\gamma k$−NN
BALANCE	0.956	0.955	0.955	0.955	0.926	0.942	0.956	0.952	**0.958**	0.952
AUTOMPG	0.828	0.845	0.832	0.838	0.858	**0.861**	0.838	0.838	0.823	0.823
IONOSPHERE	0.859	0.921	0.864	**0.925**	0.859	0.921	0.859	0.921	0.870	0.921
PIMA	0.524	0.569	0.547	0.564	0.569	0.573	0.530	**0.589**	0.525	0.557
WINE	0.882	0.894	0.896	**0.903**	0.875	0.898	0.874	0.888	0.866	0.884
GLASS	0.751	0.751	0.729	0.729	0.745	**0.772**	0.752	0.732	0.735	0.731
GERMAN	0.425	0.465	0.432	0.464	0.495	**0.505**	0.426	0.467	0.439	0.462
VEHICLE	0.888	0.881	0.893	**0.904**	0.832	0.866	0.883	0.894	0.891	0.899
HAYES	0.220	0.580	0.217	0.553	0.314	0.450	0.192	**0.593**	0.195	0.554
SEGMENTATION	0.871	0.865	0.865	0.859	0.807	0.819	0.872	0.869	**0.875**	0.870
ABALONE8	0.318	0.346	0.330	0.354	**0.373**	0.363	0.320	0.348	0.325	0.350
YEAST3	0.692	0.702	0.680	0.700	**0.717**	0.695	0.693	0.709	0.691	0.692
PAGEBLOCKS	0.852	0.859	0.852	**0.864**	0.841	0.850	0.853	0.859	0.844	0.861
SATIMAGE	0.456	0.484	0.465	0.480	0.425	**0.496**	0.454	0.484	0.468	0.484
LIBRAS	0.822	**0.838**	0.822	0.821	0.800	0.787	0.822	0.812	0.836	0.829
WINE4	0.086	0.094	0.111	**0.113**	0.071	0.096	0.086	0.094	0.092	0.094
YEAST6	0.530	0.596	0.557	0.550	0.422	0.602	0.542	0.596	0.479	**0.603**
ABALONE17	0.139	0.149	0.135	0.126	0.116	0.140	0.139	**0.170**	0.140	0.141
ABALONE20	0.094	0.101	0.093	0.052	0.070	**0.130**	0.094	0.109	0.090	0.108
MEAN	0.589	0.626	0.593	0.619	0.585	0.619	0.589	**0.628**	0.586	0.622

TABLE 6 – Results for 3−NN on the DGFiP datasets. The values correspond to the mean F-measure F_1 over 5 runs. The best result on each dataset is indicated in bold while the second is underlined.

DATASETS	3−NN	γk−NN	SMOTE	SMOTE+γk−NN
DGFIP19 2	0,454$_{(0,007)}$	<u>0,528</u>$_{(0,005)}$	0,505$_{(0,010)}$	**0,529**$_{(0,003)}$
DGFIP9 2	0,173$_{(0,074)}$	<u>0,396</u>$_{(0,018)}$	0,340$_{(0,033)}$	**0,419**$_{(0,029)}$
DGFIP4 2	0,164$_{(0,155)}$	<u>0,373</u>$_{(0,018)}$	0,368$_{(0,057)}$	**0,377**$_{(0,018)}$
DGFIP8 1	0,100$_{(0,045)}$	**0,299**$_{(0,010)}$	0,278$_{(0,043)}$	**0,299**$_{(0,011)}$
DGFIP8 2	0,140$_{(0,078)}$	0,292$_{(0,028)}$	**0,313**$_{(0,048)}$	<u>0,312</u>$_{(0,021)}$
DGFIP9 1	0,088$_{(0,090)}$	0,258$_{(0,036)}$	<u>0,270</u>$_{(0,079)}$	**0,288**$_{(0,026)}$
DGFIP4 1	0,073$_{(0,101)}$	<u>0,231</u>$_{(0,139)}$	0,199$_{(0,129)}$	**0,278**$_{(0,067)}$
DGFIP16 1	0,049$_{(0,074)}$	0,166$_{(0,065)}$	<u>0,180</u>$_{(0,061)}$	**0,191**$_{(0,081)}$
DGFIP16 2	0,210$_{(0,102)}$	0,202$_{(0,056)}$	<u>0,220</u>$_{(0,043)}$	**0,229**$_{(0,026)}$
DGFIP20 3	0,142$_{(0,015)}$	<u>0,210</u>$_{(0,019)}$	0,199$_{(0,015)}$	**0,212**$_{(0,019)}$
DGFIP5 3	0,030$_{(0,012)}$	0,105$_{(0,008)}$	**0,110**$_{(0,109)}$	<u>0,107</u>$_{(0,010)}$
MEAN	0,148$_{(0,068)}$	<u>0,278</u>$_{(0,037)}$	0,271$_{(0,057)}$	**0,295**$_{(0,028)}$

SATokE: How can Syntax-Aware Contextualized Word Representations Benefit Implicit Discourse Relation Classification?

Diana Nicoleta Popa[1], Julien Perez[2], James Henderson[3], and Eric Gaussier[1]

[1]Université Grenoble Alpes, LIG, Grenoble, France
[2]Naver Labs Europe, Meylan, France
[3]Idiap Research Institute, Martigny, Switzerland

Abstract

Automatically identifying implicit discourse relations requires an in-depth semantic understanding of the text fragments involved in such relations. While early work investigated the usefulness of different classes of input features, current state-of-the-art models mostly rely on standard pre-trained word embeddings to model the arguments of a discourse relation. In this paper, we introduce a method to compute contextualized representations of words, leveraging information from the sentence dependency parse, to improve argument representation. The resulting *token embeddings* encode the structure of the sentence from a dependency point of view in their representations. Experimental results show that the proposed representations achieve state-of-the-art results when input to standard neural network architectures, surpassing complex models that use additional data and consider the interaction between arguments.

Keywords: token embeddings, syntactic dependencies, implicit discourse relation classification

1 Introduction

Automatically identifying discourse relations is helpful for downstream NLP tasks such as question answering, machine translation or automatic summarization [DH18]. While identifying discourse relations in the presence of explicit connectives has proven to be relatively easy, with accuracy scores around 93% [PRM+08], it is more challenging to identify these relations in the absence of textual cues.

Nevertheless, much research focused on the task along with the release of the Penn Discourse Treebank (PDTB) [PDL+08], the largest annotated corpus of discourse relations, covering 2312 Wall Street Journal (WSJ) articles. In PDTB, documents are annotated following the predicate-argument structure: a discourse connective (e.g. *but, because*) is a predicate that takes two text spans around it as its arguments, further denoted as *Arg*-1 and *Arg*-2. The discourse connective is structurally attached to *Arg*-2 and can be either explicit or implicit.

An example of an implicit discourse relation is provided below: the implicit connective "so" is not part of the original argument text but is added to illustrate the idea.

- ***Arg*-1:** "A lot of investor confidence comes from the fact that they can speak to us," he says.
  ***Arg*-2:** [**so**] "To maintain that dialogue is absolutely crucial."
  Class: Cause

To approach the task of implicit discourse relation classification (IDRC), the focus of most work has been on modelling the interaction between arguments and less on their representations. Although earlier work uses different feature sets as input: word pairs, part-of-speech tags, context information etc. [PLN09], little attention has been offered to varying the input features of recent deep neural network-based approaches to IDRC and to how these can influence the output quality of such models. That is, most current approaches rely on standard pre-trained word type embeddings, which have been successful across a variety of tasks [ZSCM13, TQM15] and have been proven to perform best so far on IDRC as well [BD15].

To account for the difficulty to fully recover the semantics of arguments using only surface level features [JE15], additional linguistic and structural information regarding the arguments can be leveraged [DH18, QZZ16a]. However, information from syntactic dependencies, previously proven beneficial for the task [LKN09], is not explicitly integrated in any of these models. The current work aims precisely at investigating the use of syntactic information in such a context.

On the other hand, there has been a strong recent interest to replace the generic word type embeddings by *token embeddings* which have proven to be successful within various applications [MBXS17, PNI+18]. The idea of token embeddings relies on representing a word in its context, with the same word bearing different representations in different contexts. This contrasts to a generic word type embedding representation, which is the same in every context.

Since the task of detecting implicit discourse relations requires semantic understanding which consequently relies

on encoding the word meaning in its context [QZZ16a], it is natural to investigate the use of token representations for this task. For this we propose a set of token representations that offer improvements over traditional word representations and have the benefit of encoding the information about the structure of the sentence from a dependency point of view in the representations themselves. To the best of our knowledge we are the first to investigate the use of *token embeddings* for the task of IDRC and to analyze the impact of using syntactic dependencies information as input to deep learning models for this task.

We evaluate the proposed token embeddings as input to two most common basic neural network architectures and an additional gated mechanism to model the interaction between the arguments. The main contributions of the current work are:

- We propose a method that explicitly integrates syntactic information into token embeddings to model the arguments for IDRC;
- We analyze and compare the contribution of token embeddings to the task as opposed to word type representations;
- We reach state-of-the-art performance with simple architectures, surpassing complex models that focus on the interaction between arguments or use additional data as well as models that use other token embeddings.

The remaining of this paper is organised as follows: We first present the related work in Section 2, then we introduce our token embeddings proposal in Section 3.1, the architectures used to model the arguments in a discourse relation in Section 3.2 and the methods for discourse relation identification in Section 3.3. Details about the data used and the implementation are provided in Section 4. Section 5 presents the results along with an analysis of the proposed contribution.

2 Related work

2.1 Implicit discourse relation classification

The task of IDRC is typically approached as a classification problem, with the two arguments as input and their implicit discourse relation as the label to predict. Some work leverages additional information from unlabeled data by removing the existing explicit connectives [RX15]. However, it has been proven that the nature of the natively implicit data may be different from that of an artificially constructed implicit relation corpus from data with explicit connectives [LKN09]. Other work leverages both labeled and unlabeled data (implicit and explicit) from different corpora in order to classify discourse relations in multi-task learning frameworks [LLZS16]. Another tendency is to focus on the interaction between the two arguments either through attention mechanisms [LWW+17], by using features derived from word pairs [CZL+16] or by modelling the argument pair jointly [LLZS16].

Regardless of the chosen approach, accurately representing the arguments is key to building a reliable model. For this, most work relies on standard word embeddings to model the arguments of a discourse relation [QZZ+17, WLY+17, DH18]. Some work employs complementary features to integrate additional knowledge: [JE15] represent each argument using bottom-up compositional operations over its constituency parse tree along with a top-down approach for modelling coreferent entity mentions. Others complement word embeddings with extra linguistic features like part-of-speech tag embeddings [DH18] or character-level information [QZZ16a]. [BD16] learn distributional word representations tailored specifically for IDRC.

We propose to improve the representation of words by using *token embeddings* computed using dependency information. Dependency information has been shown to be beneficial for detecting implicit discourse relations in traditional models [LKN09]. However, this information is not explicitly used in more recent neural architectures. We separate the token embeddings computation from the task at hand which enables assessing the benefits of using them in comparison to standard word type embeddings.

2.2 Token embeddings methods

There has been a growing interest recently in representing text using token embeddings. [DADH17] propose a method to obtain context-aware token embeddings by estimating a distribution over semantic concepts (synsets) extracted from WordNet. [TGL17] use a feed forward neural network to produce token embeddings which are further evaluated as features for part-of-speech taggers and dependency parsers.

[MBXS17] provide context-aware vectors (CoVe) by transferring a pre-trained deep LSTM encoder from a model trained for machine translation to a variety of other NLP tasks. Recent work leverages information from language models in semi-supervised settings: [PABP17] use the parameters from a pre-trained language model to induce contextual information to token representations. [PNI+18] extend the idea by learning a task specific linear combination of the intermediate layers of a deeper bidirectional language model (ELMo). However, their representation is somewhat task dependent in that the linear combination is learned with respect to the task at hand. Training these encoders requires though large amounts of data. [DCLT19] also provide a method to obtain token embeddings by leveraging an architecture based on multi-layer bidirectional Transformers [VSP+17].

A parallel could be drawn between the current proposal and the work of [SB18] who show that adding contextualized representations to a basic model for question-answering achieves state-of-the-art results, despite using only minimal question-document interaction. This validates further the importance of having contextually-informed fea-

tures as input to deep learning models, even when these models are not the most complex.

3 Main approach

We propose to approach the task of IDRC through a two-step process: unsupervised computation of syntactically-aware contextualized representations of words and a supervised model for the prediction of discourse relations. The proposed token embeddings will constitute an informed and complete encoding as they are trained to predict the relations holding between them in the sentence graph.

3.1 Computing the token embeddings

We compute unsupervised token embeddings for all the words in the corpora. All sentences are parsed and further encoded using the method described further. The dependency information used in the current work is obtained with the spaCy toolkit [1], leveraging the transition-based dependency parser of [HJ15] and using the CLEAR label scheme for the syntactic dependencies. We additionally use information regarding immediate local context in the form of adjacency relations.

Formally, given the graph of a sentence G_s as provided by its dependency parse tree, we model the interactions between the tokens in the sentence as shown in Figure 1. We use a rank-3 tensor T_s to specify the binary relations between tokens that are given by the parse tree of the sentence along with an additional adjacency relation.

Given the dependency parse of a sentence s, the tensor T_s is formed of r matrices: $r-1$ matrices corresponding to information from $r-1$ syntactic dependency relations (as provided by the parser) and 1 matrix corresponding to information regarding the adjacency relation. Each of these matrices is of dimensions $|s| \times |s|$. Different matrices are used to model the different interaction types between the tokens in the sentence. An entry t^s_{ijk} in tensor T_s takes value 1 if the token at position i is in the dependency relation k with the token at position j such that the token at position i is the head of that relation and the token at position j is the dependant. All the remaining entries are set to value 0. The matrix corresponding to the adjacency information is populated similarly with value 1 whenever the token at position j immediately follows the token at position i in the given sequence and 0 otherwise.

The goal is, given the tensor T_s of a sentence s, to obtain a token embedding for each token in the sentence as well as a relation embedding for each relation that holds between tokens. While the set of token embeddings is different for each sentence, the embeddings of relations are shared across sentences. To achieve this, we optimise a ranking loss within the tensor (T^s_{loss}) that aims at scoring positive triples $t^s_{ijk} = (i, k, j)$ higher than negative ones. An additional regularisation term (R^s_{loss}) is used to minimise the gap between the token embeddings representation and a pre-trained word type representation of the words they denote [2]. This is conceptually similar to the vector space preservation term in [MÓST+16] in that it controls for how much the token embeddings can deviate from their corresponding word representations.

Our overall goal is to create embeddings that are close, through R^s_{loss}, to the original word embeddings (known to capture semantics) and at the same time are syntactically-informed, through T^s_{loss}, so as to capture fine-grained semantic differences according to the role a given word plays in a sentence. It is thus important to stress that optimising only T^s_{loss} would be insufficient as it would lack the notion of semantics provided through R^s_{loss}.

The optimisation problem is formulated as

$$\min \sum_{s \in S} \alpha(T^s_{loss}) + (1-\alpha)(R^s_{loss}) \tag{1}$$

where:

$$T^s_{loss} = \sum_{\substack{t^s_{ijk} \in G_s, \\ t^s_{i'j'k'} \in \neg(t^s_{ijk})}} \max(0, \gamma + \langle e^s_{i'}, R_{k'}, e^s_{j'} \rangle - \langle e^s_i, R_k, e^s_j \rangle)$$

$$R^s_{loss} = \sum_{e^s_i \in G_s} -\log \sigma(e^s_i \cdot w^s_i)$$

with G_s representing the graph of sentence s and holding all tokens and all relations present in the sentence, e^s_i being the token embedding of token i in sentence s, R_k the matrix embedding for the relation k, w^s_i the pre-trained word type embedding corresponding to the token e^s_i, γ the margin hyperparameter, σ the sigmoid function, and $\neg(t^s_{ijk})$ the set of negative triples associated with t_{ijk}.

E_s is the matrix holding all token embeddings for sentence s, with one token embedding per row, and R is the tensor of all relations. Given all tokens in one sentence s, that results in approximating the relations tensor for sentence s, T_s by $E_s^{(|s| \times d)} \cdot R^{(d \times r \times d)} \cdot E_s^{T(d \times |s|)}$.

The regularisation term R^s_{loss} can be seen as a particular case of cross entropy

$$R^s_{loss} = -y \cdot \log \hat{y} - (1-y) \cdot \log(1-\hat{y})$$

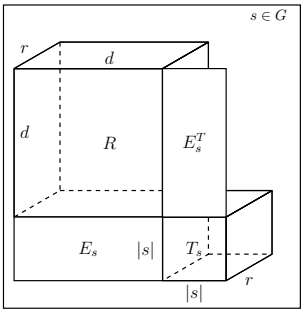
with $y = 1$ and $\hat{y} = \sigma(e^s_i \cdot w^s_i)$.

A negatively sampled example in the tensor is obtained by altering one element of the triple while fixing the remaining two as it is standard practice in such factorization methods [BUGD+13] : this element can be either one of the entities or the relation holding between them. As previously mentioned, given a triple $t^s_{ijk} = (i, k, j)$, we denote by $\neg(t^s_{ijk})$ the set of negative examples associated to it. We consider here that $\neg(t^s_{ijk})$ is formed of the following elements:

$$\neg(t^s_{ijk}) = \{(i', k, j), (i, k', j), (i, k, j')\}$$

[1] https://spacy.io/

[2] In the current work we use GloVe [PSM14] pre-trained word type embeddings to constrain the semantics of the token embeddings.



(a)

G	corpus of sentence graphs
$T_s : r \times \lvert s\rvert \times \lvert s\rvert$	relations between tokens in sentence $s \in G$
$R : r \times d \times d$	relation tensor embeddings
$E_s : \lvert s\rvert \times d$	token vector embeddings of tokens in $s \in G$
d	size of the embedding vectors
r	size of binary relations
$\lvert s\rvert$	number of tokens in sentence s

$\forall s,\ T_s \approx E_s \cdot R \cdot E_s^T$

(b)

Figure 1: Sentence graph decomposition

$\forall i' \neq i, j' \neq j, k' \neq k$.

We optimise Eq.(1) using mini-batch stochastic gradient descent to optimise all R_k and e_i^s. In the current proposal we aim to learn a representation for each word token in each sentence as well as for each relation holding between these tokens from scratch. Alternatively, one could leverage pre-trained relations embeddings - a setup whose exploration we leave for future work.

We additionally note that the method described can be applied to any other dataset of sentences provided one has access to parsing information and pre-trained word type embeddings. The obtained representations can be used to predict discourse relations as described further.

3.2 Modelling the discourse arguments

Let us consider the two arguments *Arg*-1 and *Arg*-2 with lengths n and m respectively. We associate each word w with a vector representation $\boldsymbol{x}_w \in \mathbb{R}^d$. Let $\boldsymbol{x}_i^1$ and $\boldsymbol{x}_i^2$ be the d-dimensional i-th word vector in *Arg*-1 and *Arg*-2 respectively. Then the word representations of the two arguments are: *Arg*-1 $= [\boldsymbol{x}_1^1, \boldsymbol{x}_2^1, \ldots, \boldsymbol{x}_n^1]$ and *Arg*-2 $= [\boldsymbol{x}_1^2, \boldsymbol{x}_2^2, \ldots, \boldsymbol{x}_m^2]$

3.2.1 Recurrent architecture

One of the most widely used models to encode sequences, that addresses the issue of learning long-distance dependencies and that takes into account contextual information, is the long-short term memory LSTM [HS97], a variant of the recurrent neural network. We adopt two LSTM neural networks to model the two arguments separately. Given a word sequence representation $[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k]$ as input, an LSTM computes the hidden state sequence representation $[\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_k]$. At each time step i, the model reads w_i as input and updates the h_i hidden state, according to a set of transformations: $\boldsymbol{i}_i = \sigma(\boldsymbol{W}^{xi}\boldsymbol{x}_i + \boldsymbol{W}^{hi}\boldsymbol{h}_{i-1} + \boldsymbol{b}^i)$; $\boldsymbol{f}_i = \sigma(\boldsymbol{W}^{xf}\boldsymbol{x}_i + \boldsymbol{W}^{hf}\boldsymbol{h}_{i-1} + \boldsymbol{b}^f)$; $\boldsymbol{o}_i = \sigma(\boldsymbol{W}^{xo}\boldsymbol{x}_i + \boldsymbol{W}^{ho}\boldsymbol{h}_{i-1} + \boldsymbol{b}^o)$; $\boldsymbol{c}_i = \boldsymbol{f}_i \odot \boldsymbol{c}_{i-1} + \boldsymbol{i}_i \odot \tanh(\boldsymbol{W}^{xc}\boldsymbol{x}_i + \boldsymbol{W}^{hc}\boldsymbol{h}_{i-1} + \boldsymbol{b}^c)$; $\boldsymbol{h}_i = \boldsymbol{o}_i \odot \tanh \boldsymbol{c}_i$, with $\odot$ representing the element-wise product and σ the sigmoid function. $\boldsymbol{i}$, $\boldsymbol{f}$, $\boldsymbol{o}$ are the input gate, forget gate and output gate respectively and $\boldsymbol{c}$ is the memory cell. $\boldsymbol{W}^{xi}$, $\boldsymbol{W}^{xf}$, $\boldsymbol{W}^{xo}$, $\boldsymbol{W}^{hi}$, $\boldsymbol{W}^{hf}$, $\boldsymbol{W}^{ho}$, $\boldsymbol{W}^{xc}$, $\boldsymbol{W}^{hc}$, $\boldsymbol{b}^i$, $\boldsymbol{b}^f$, $\boldsymbol{b}^o$, $\boldsymbol{b}^c$ are parameters of the model. The final representations for each of the two arguments is then given by the last hidden state representation for each of them: *Arg*-1 $= \boldsymbol{h}_n$ and *Arg*-2 $= \boldsymbol{h}_m$.

3.2.2 Convolutional architecture

In order to represent each argument using convolutional neural networks (CNNs), we follow the approach of [Kim14]. Given a word sequence representation $[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k]$, let $\boldsymbol{W} \in \mathbb{R}^{h \times d}$ be a filter applied to a window of h words to produce a feature c_i, b a bias term and f a non-linear function. Then $c_i = f(\boldsymbol{W} \cdot \boldsymbol{x}_{i:i+h-1} + b)$. A feature map $\mathbf{c} \in \mathbb{R}^{n-h+1}$ is created by applying the filter to each possible window of words in the argument $\{\boldsymbol{x}_{1:h}, \boldsymbol{x}_{2:h+1}, \ldots, \boldsymbol{x}_{k-h+1:k}\}$. Thus $\mathbf{c} = [c_1, c_2, ..., c_{k-h+1}]$, followed by a max-pooling operation $\hat{c} = max(\mathbf{c})$ to obtain the most important feature, the one of highest value, corresponding to the particular filter. For m filters with different window sizes we obtain m features: $z = [\hat{c}_1, ..., \hat{c}_m]$. The representation of each argument is the m-dimensional vector $\boldsymbol{z}$: *Arg*-1 $= \boldsymbol{z}_1$ and *Arg*-2 $= \boldsymbol{z}_2$.

3.3 Predicting the discourse relation

The discourse relation holding between two arguments can be predicted with or without modelling the interaction between the two arguments, as presented in the following.

No interaction between arguments Once we obtain the vector representations of each argument (through LSTM or CNN), we concatenate these vectors into a vector of the pair $\boldsymbol{v} = [\boldsymbol{h}_n, \boldsymbol{h}_m]$ for an LSTM encoding and $\boldsymbol{v} = [\boldsymbol{z}_1, \boldsymbol{z}_2]$ for a CNN encoding. The representation of the pair is further passed to a fully connected layer, followed by a softmax layer to obtain the probability distribution over labels. This approach, however, does not focus on modelling the interaction between the two arguments in the discourse relation,

an important aspect when predicting discourse relations, as pointed out by previous work [CZL+16, LWW+17].

Collaborative gated neural network An alternative approach that does enable modelling argument interaction (to some extent) is the collaborative gated neural network (CGNN) [QZZ16b]. In CGNN the arguments are modelled using CNNs that share parameters and an additional gated unit is used for feature transformation. The input to the CGNN unit is the vector of the pair $\boldsymbol{v} = [\boldsymbol{z}_1, \boldsymbol{z}_2]$ and the set of transformations are: $\hat{c} = \tanh(\boldsymbol{W}^c \cdot \boldsymbol{v} + \boldsymbol{b}^c)$; $\boldsymbol{g}_i = \sigma(\boldsymbol{W}^i \cdot \boldsymbol{v} + \boldsymbol{b}^i)$; $\boldsymbol{g}_o = \sigma(\boldsymbol{W}^o \cdot \boldsymbol{v} + \boldsymbol{b}^o)$; $\boldsymbol{c} = \hat{c} \odot \boldsymbol{g}_i$; $\boldsymbol{h} = \tanh(\boldsymbol{c}) \odot \boldsymbol{g}_o$, where $\odot$ denotes the element-wise multiplication, σ denotes the sigmoid function, $\hat{c}$ and $\boldsymbol{c}$ are inner cells, $\boldsymbol{g}_i$ and $\boldsymbol{g}_o$ are the two gated operations and $\boldsymbol{W}^i$, $\boldsymbol{W}^o$ and $\boldsymbol{W}^c$ are parameters of the model. The output of the CGNN unit is the transformed vector $\boldsymbol{h}$ which is further passed to a softmax layer.

We define the training objective as the cross-entropy loss between the output of the softmax layer and the class labels.

4 Experimental setup

4.1 Data

Throughout all experiments we use the PDTB 2.0 dataset [PDL+08]. The dataset contains a total of 16224 argument pairs annotated with their corresponding explicit and implicit discourse connectives at three levels of granularity: class, type and subtype. Level-1 contains four semantic classes: Comparison, Contingency, Expansion and Temporal, whereas level-2 contains 16 types that refine the relation senses to provide finer semantic distinctions. Level-3 is typically ignored in the literature as it provides too fine-grained subtypes and is not present for all the types [LKN09].

In all the experiments only argument pairs annotated with *implicit* discourse relations are considered and data marked with explicit connectives is not leveraged in any way. To enable comparisons with previous work, we follow two popular experimental setups and perform multi-class classification on both level-1 and level-2.

We adopt the same split as in [LKN09], further denoted as PDTB-Lin and perform multi-class classification for level 2 classes. Similarly to [LKN09], the least frequent 5 types are removed as they account for only 9 examples in the training data and no examples in the development and test sets. In PDTB about 2.2% of the implicit relations are annotated with two types. Following previous work, during training, instances with more than one annotation are considered as multiple instances, each with one type annotation. During testing, a correct prediction is one that matches one of the annotated types.

A second split is that of [PLN09] denoted PDTB-Pitl, for which we report results for level-1 multi-class classification similarly to previous work.

4.2 Implementation details

For token embeddings computation we run multiple threads varying the initial learning rate, the negative sampling factors and the margin γ. We fix the ratio between the two losses to $\alpha = 0.5$, optimise using Adam [KB14] and do early stopping based on the validation loss. All token embeddings are randomly initialised [3]. We obtain the best results when using an initial learning rate of 10^{-3}, a sampling factor of 5x and γ set to 10. We consider all dependency relations with a frequency higher than 1000 in the corpus. All relations with frequency lower are still considered, but they are all merged under the "unknown_rel" tag.

A distribution of the dependency relations considered for the task is presented in Figure 2. The most frequent relations present in the dataset are: prep (preposition), pobj (object of preposition), det (determiner), compound and nsubj (nominal subject). About one third of all relations have frequencies above the average, while the "unknown_rel" relation represents a very low proportion of the data.

There is a total of 595 629 tokens in the corpus, covering the occurrences of 30 554 unique word types. For about 2.3% of the tokens, a corresponding GloVe embedding was not available, which resulted in replacing such tokens with a generic "unknown_POS" tag where "POS" stands for the part of speech of the given token. Representations are thus learned for these tokens as well as for the "unknown_POS" word types they denote. As at this stage SATokE does not leverage character level information, representations of such out-of-vocabulary words should indeed be more accurate the more data is considered.

For the optimisation of the implicit discourse relation classification task we use Adam [KB14] and the parameters of the models are tuned on the development set. The parameters that lead to the reported results for each split are shown in Table 1, with α_{LR} denoting the initial learning rate, dp the dropout probability, nbh the size of the fully connected layer, szb the size of the batches and nbf the number of filters. The parameters of the CNN are shared across the two arguments and the filters used are of sizes 3, 4, 5.

5 Results and discussion

We present a comparison of the proposed token embeddings to standard word type embeddings and to the state-of-the-art token embeddings method ELMo. A parallel is then drawn between our results and state-of-the-art systems and a series of experiments are presented with model variations. All word type embeddings considered are 300-dimensional. The SATokE token embeddings proposed in the current work are 300-dimensional as well, while the ELMo token embeddings are 1024-dimensional. The considered ELMo representations are obtained as a weighted sum of the different layers of the deep bidirectional lan-

[3] Initialization with the pre-trained GloVe [PSM14] word type embeddings did not result in improvements.

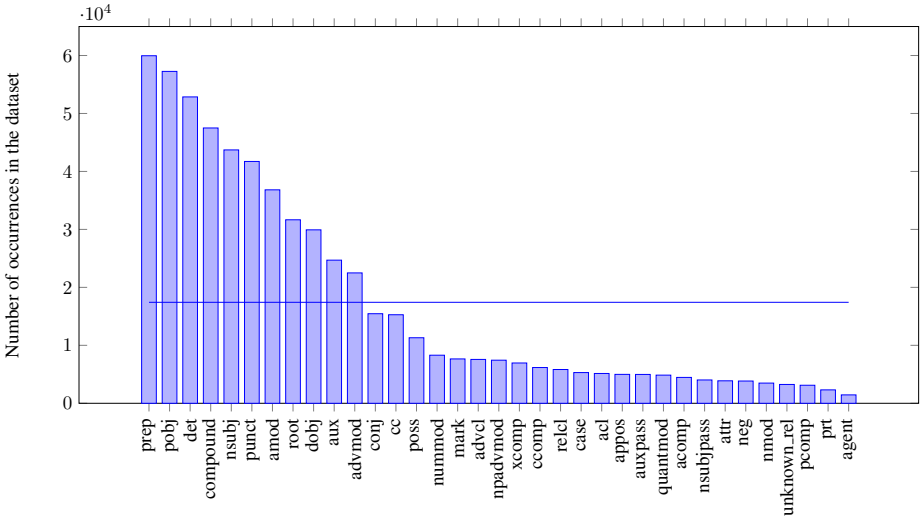


Figure 2: Distribution of the considered syntactic relations in the PDTB data.

Data split	LSTM runs			CNN runs					CGNN runs				
	α_{LR}	dp	nbh	α_{LR}	dp	nbf	szb	nbh	α_{LR}	dp	nbf	szb	nbh
PDTB-Lin	10^{-4}	0.7	150	10^{-5}	0.8	600	16	300	10^{-4}	0.7	128	64	768
PDTB-Pitler	10^{-5}	0.7	512	10^{-5}	0.6	600	16	512	10^{-4}	0.6	600	64	3600

Table 1: Parameters used for the reported results on each data split

guage model, with the weights being trainable. This is supported by the claim of [PNI+18] that the semantic and syntactic aspects are captured in different intermediate layers and including representations from all layers improves overall performance. In that sense, combining the intermediate layers for ELMo represents a trade-off between the two aspects. In the case of SATokE, the semantic and syntactic aspects are weighted equally given the ratio between the two loses is set to $\alpha = 0.5$.

5.1 Comparison to word type embeddings

Table 2 presents the results of three sets of experiments on the PDTB-Lin data, using different input features: standard pre-trained word type embeddings often employed in literature [PSM14, MSC+13], word type embeddings trained using dependency contexts [LG14] and our proposed syntactically-aware token representations (SATokE).

Using the proposed token embeddings as input yields important improvements over all word type embeddings we compare to, across all architectures considered. We obtain improvements between 1.5% and 4.5% when using an LSTM architecture, 3.7% to 7.5% with a CNN encoder and 4% to 8% with CGNN. Unsurprinsingly, feeding SATokE as input to an LSTM encoder only improves results by up to 4.5%. This could be explained by the fact that SATokE encode positional information by their construction using adjacency information, and thus they complement less the advantages of using an LSTM encoder. Alternatively, the higher difference, with respect to the CNN results, observed in the case of SATokE (as opposed to the differences observed for the word type embeddings) could be due to the fact that each argument in the pair is represented only by the last hidden state of the LSTM that is further passed to subsequent layers, while the CNN architecture has access to the computed token embeddings at each position.

	LSTM	CNN	CGNN
GloVe 300d	38.97	38.25	39.03
Word2Vec 300d	36.92	37.33	37.07
Deps-WE 300d	36.00	34.98	34.98
SATokE 300d	**40.51**	**42.55**	**43.08**

Table 2: Results for level-2 classification on PDTB-Lin

Embeddings trained on dependency contexts yield consistently worse results than all other embeddings considered. This is consistent with [GFEC16] who show that such embeddings provide decreased perfomance on semantic tasks despite high performance on POS tagging or chunckking. Contrary to a first interpretation, dependency information can be useful for semantic tasks: its value is higher when this information is injected into the token representations directly, by leveraging the parse tree of the sentence, than when used as context for creating generic word embeddings from a large corpus. The observed results support this statement with improvements of 4.5% to 8% in absolute accuracy between Deps-WE and SATokE.

Model	Accuracy (%)
[LKN09]	40.2
[QZZ16a]	43.81
[QZZ+17]	44.65
CGNN+SATokE	43.08

Table 3: Comparison to related work on level-2 PDTB-Lin.

5.2 Comparison to related work

Table 3 compares the previously presented results to the related work for the fine-grained multi-class classification of the PDTB-Lin split. Although the scores obtained using the SATokE token embeddings are close to those of state-of-the-art systems, it is important to note that they either use additional information from non-implicit relations to train their models or consider more complex architectures:

[LKN09] exploit and analyze the use of features derived from constituent and dependency parse trees as well as word pair features as input to a traditional maximum entropy classifier. [QZZ16a] use a hybrid architecture in which they enhance the word embeddings representations with character level information derived from stacked convolutional and bidirectional LSTM layers. [QZZ+17] use a more complex model in an adversarial framework leveraging explicit discourse connectives. Their adversarial setup is composed of a network fed only data with implicit relations, a network fed data augmented with connectives, a discriminator aiming at distinguishing between the features produced by the two and a final classifier to determine the discourse relation between the two arguments.

To enable further comparisons to more complex models, we run a set of experiments on level-1 multi-class classification on the PDTB-Pitl split, which we compare to related work in Table 4. While the results obtained are competitive to related work, the systems we compare to differ from the current proposal from several points of view. Some work uses additional data from different corpora or containing explicit connectives [RX15, LLZS16, JHE16, LWW+17, DH18]. Other work focuses on complex architectures to model the interaction between the two arguments [LLZS16, LL16, JHE16, LWW+17, DH18].

Further details about each work are provided in the following: [RX15] collect additional training data through distant supervision over freely omissible explicit discourse connectives. [LLZS16] use a multi-task neural network framework leveraging a combination of different discourse corpora. They use CNNs to represent the pairs of arguments, creating both unique and shared representations across the different tasks and consider additional surface-level features. [LWW+17] use an attention-based LSTM to model the interaction between the two arguments integrated in a multi-task joint learning setting. They also leverage explicit discourse relations and unlabelled external data.

[LL16] use a multi-level attention mechanism to repeatedly read the arguments involved in a discourse relation along with an external short-term memory to keep track of the exploited information. The arguments are modeled using bidirectional LSTM networks. [DH18] model interdependencies between arguments and the sequence patterns of their discourse connectives by positioning them in the wider context of paragraphs. [JHE16] model jointly the discourse relation and the arguments with a latent variable RNN-based architecture using additional data from non-implicit relations. Their proposed probabilistic model is used to infer discourse relations given the representations of the arguments as well as to do discourse-aware language modelling. [WLY+17] focus on the representation of the arguments: they propose Tree-LSTM and Tree-GRU models to encode the arguments. They use the structure of the constituency parse trees to recursively compose semantics bottom-up and constituent tags to control for semantic composition and induce grammatical information.[5]

However, none of these investigates the use of syntactic dependencies or can encode arbitrary graphs. We observe that by using a simple CNN encoder with SATokE as input, we obtain state-of-the-art results, even surpassing most complex models which exploit external data, have a higher number of parameters and/or model the interaction between arguments. It is important to note that out of the models that use neither additional data nor argument interaction, SATokE yields the best results, above ELMo (Peters et al. 2018), a state-of-the-art token embedding method. Also the CNN+SATokE setup represents the one with the lowest number of parameters for the given performance level. The use of CNN in SATokE yields better results than the use of CGNN, suggesting that the CGNN architecture may not constitute a powerful enough architecture for this setup.

5.3 Impact of syntax - model variations

While modelling the words in their context seems beneficial, we want to further investigate to what extent syntax plays an important role. In Table 5, an additional set of experiments analyze the impact of using dependency information in the computation of the token embeddings with a CNN architecture. We first set the parameters for the token embeddings computation to the ones that obtained the best results in the default scenario, further denoted as **SATokE** in Table 5. Then we consider two comparative settings: token embeddings computed without the adjacency relation *SATokE-adj*, and without all syntactic relations *SATokE-syntax* respectively. The decrease of performance in results shows that both adjacency relation and syntax play an important role in the final result. However, the results seem to degrade more when information about syntax is removed from the token computation than when adjacency information is not present.

Further experiments take into account only certain types of dependency relations considered to be highly relevant.

[5] The reproduction of the exact scores reported by [WLY+17] was impossible, despite re-creating the setup described in the paper with the code provided by the authors.

Model	Accuracy (%)	Additional data	Arg Interaction	Multi-task	$\|\theta\|$
[LLZS16]	*57.27*	✓	✓	✓	$> 1M$
[LWW+17]	*57.39*	✓	✓	✓	$> 1M$
[LL16]	*57.57*	-	✓	-	$6.7M$
[DH18]	*58.2*	✓	✓	-	$> 4M$
[JHE16]	*59.5*	✓	✓	-	$5.5M$
[WLY+17]	56.04	-	-	-	$6.7M$
LSTM + GloVe 300d	54.60	-	-	-	$1.5M$
CNN + GloVe 300d	56.42	-	-	-	$4M$
LSTM + ELMo 1024d	56.97	-	-	-	$1.5M$
CNN + ELMo 1024d	57.81	-	-	-	$4M$
LSTM + SATokE 300d	56.63	-	-	-	$1.5M$
CNN + SATokE 300d	**58.83**	-	-	-	$4M$
CGNN + SATokE 300d	57.90	-	✓	-	$41M$

Table 4: Comparison to related work on level-1 PDTB-Pitl. In italic and above the double line are reported scores. All non-italic scores below the double line are (re)produced in the current work. [4]

The *SATokEAdj+SUBJ+OBJ+MOD+1rel* setup refers to tokens computed taking into account information derived from the SUBJ, OBJ and MOD dependency relations along with adjacency information. In this setting, one additional relation stands for all the other dependency relations that occur between the tokens with frequency higher than the previously set threshold. This additional relation is modelled by one single matrix in the tensor. Consequently, this reduces the sparsity as it aggregates information that would otherwise be at the basis of several matrices. The resulting tensors have thus less relations, also making the learning process faster. It can be observed that the scores obtained in these settings are not far from the best obtained scores overall. This may indicate that selecting only certain dependency relations to use and merging all the rest, could provide good results, while at the same time reducing the computation time and fighting the data sparsity issue when creating the token embeddings.

The two settings *fine-grained* and *coarse-grained* denote whether the original distinctions have been maintained or not for the more granular dependency relations. For example in the *fine-grained* setting, the nsubj (nominal subject), nsubjpass (nominal subject passive), csubj (clausal subject) and csubjpass (clausal subject passive) relations are all kept separate and thus constitute separate matrices in the tensor of a sentence, while in the *coarse-grained* setting they are all merged into the SUBJ relation and modelled using a single matrix in the tensor. It can be observed from Table 5 that using the *coarse-grained* setup yields a 0.5% increase over its corresponding *fine-grained* counterpart. This can be related to the sparsity issue: as Figure 2 shows, there are not enough examples of lower granularity dependency relations in the corpus to account for having a separate matrix for each of these relations and to drive efficient learning: see the low frequencies of nmod (modifier of nominal), nsubjpass (nominal subject passive) and quantmod (modifier of quantifier). Thus having a single general relation that stands for multiple fine-grained relations could be beneficial in the lack of high amounts of data. Overall, these results also suggest that there may be settings in which other dependency relations could be merged to obtain better results.

Finally, we consider a set of experiments in which we iteratively remove certain dependency relations considered important, from the token embeddings computation. We obtain tokens computed without information coming from the SUBJ, OBJ and MOD dependency relations. We observe that, for the most part, results degrade the more information is removed from the computation of the tokens.

6 Conclusion

The task of implicit discourse relation classification requires an in-depth understanding of the arguments involved in a discourse relation. To tackle this challenging aspect, we propose to explicitly inject syntactic information to create contextualized word representations, SATokE. We show that the proposed embeddings outperform standard pre-trained word type representations as well as state-of-the-art token embeddings for this task. We additionally show that using simple neural network architectures, we can integrate the proposed representations into a model for implicit discourse relation classification that achieves state-of-the-art results without additional data and without modelling the interaction between arguments.

References

[BD15] Chloé Braud and Pascal Denis. Comparing word representations for implicit discourse

[5] Estimations for parameter count are provided whenever the details in the corresponding work are not sufficient to compute an exact number. All numbers assume an input embedding size of 300.

Model variations	Accuracy (%)
Original model	
SATokE	**58.83**
Removing information	
SATokE-adj	57.81
SATokE-syntax	57.65
Limited information	
SATokEAdj+SUBJ+OBJ+MOD+1rel_fine-grained	58.07
SATokEAdj+SUBJ+OBJ+MOD+1rel_coarse-grained	58.57
Filtering	
SATokE-SUBJ	58.83
SATokE-SUBJ-OBJ	57.90
SATokE-SUBJ-OBJ-MOD	57.56

Table 5: Results for level-1 classification on PDTB-Pitl using variations of the proposed token embeddings model.

relation classification. In *Proceedings of EMNLP*, 2015.

[BD16] Chloé Braud and Pascal Denis. Learning connective-based word representations for implicit discourse relation identification. In *Proceedings of EMNLP*, 2016.

[BUGD+13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*. 2013.

[CZL+16] Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Implicit discourse relation detection via a deep architecture with gated relevance network. In *Proceedings of ACL*, 2016.

[DADH17] Pradeep Dasigi, Waleed Ammar, Chris Dyer, and Eduard Hovy. Ontology-aware token embeddings for prepositional phrase attachment. In *Proceedings of ACL*, 2017.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL HLT*, 2019.

[DH18] Z. Dai and R. Huang. Improving Implicit Discourse Relation Classification by Modeling Inter-dependencies of Discourse Units in a Paragraph. *ArXiv e-prints*, 2018.

[GFEC16] Sahar Ghannay, Benoit Favre, Yannick Estève, and Nathalie Camelin. Word embeddings evaluation and combination. In *Proceedings of JLRE*, 2016.

[HJ15] Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In *Proceedings of EMNLP*, 2015.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[JE15] Yangfeng Ji and Jacob Eisenstein. One vector is not enough: Entity-augmented distributed semantics for discourse relations. In *TACL*, 2015.

[JHE16] Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of NAACL HLT*, 2016.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2014.

[Kim14] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, 2014.

[LG14] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of ACL*, 2014.

[LKN09] Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of EMNLP*, 2009.

[LL16] Yang Liu and Sujian Li. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. In *Proceedings of EMNLP*, 2016.

[LLZS16] Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. Implicit discourse relation classification via multi-task neural networks. In *Proceedings of AAAI*, 2016.

[LWW+17] Man Lan, Jianxiang Wang, Yuanbin Wu, Zheng-Yu Niu, and Haifeng Wang. Multi-task attention-based neural networks for implicit discourse relationship representation and identification. In *Proceedings of EMNLP*, 2017.

[MBXS17] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Proceedings of NIPS 30*, 2017.

[MÓST+16] Nikola Mrkšic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašic, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL HLT*, 2016.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS 26*, 2013.

[PABP17] Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of ACL*, 2017.

[PDL+08] Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. The Penn Discourse TreeBank 2.0. In *Proceedings of LREC*, 2008.

[PLN09] Emily Pitler, Annie Louis, and Ani Nenkova. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of ACL*, 2009.

[PNI+18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL HLT*, 2018.

[PRM+08] Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind Joshi. Easily identifiable discourse relations. *Proceedings of COLING*, 2008.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, 2014.

[QZZ16a] Lianhui Qin, Zhisong Zhang, and Hai Zhao. Implicit discourse relation recognition with context-aware character-enhanced embeddings. In *Proceedings of COLING*, 2016.

[QZZ16b] Lianhui Qin, Zhisong Zhang, and Hai Zhao. A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of EMNLP*, 2016.

[QZZ+17] Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of ACL*, 2017.

[RX15] Attapol Rutherford and Nianwen Xue. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *Proceedings of NAACL-HLT*, 2015.

[SB18] Shimi Salant and Jonathan Berant. Contextualized word representations for reading comprehension. In *Proceedings of NAACL HLT*, 2018.

[TGL17] Lifu Tu, Kevin Gimpel, and Karen Livescu. Learning to embed words in context for syntactic tasks. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017.

[TQM15] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21st ACM SIGKDD*, 2015.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*. 2017.

[WLY+17] Yizhong Wang, Sujian Li, Jingfeng Yang, Xu Sun, and Houfeng Wang. Tag-enhanced tree-structured neural networks for implicit discourse relation classification. In *Proceedings of IJCNLP*, 2017.

[ZSCM13] Will Y. Zou, Richard Socher, Daniel M. Cer, and Christopher D. Manning. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*, 2013.

On learning a large margin classifier for domain adaptation based on similarity functions

Sofien Dhouib[1], Ievgen Redko*[2], et Carole Lartizien†[1]

[1]Univ Lyon, INSA-Lyon, Université Claude Bernard Lyon 1, UJM-Saint Etienne, CNRS, Inserm, CREATIS UMR 5220, U1206, F-69100, LYON, France
[2]Univ Lyon, UJM-Saint-Etienne, CNRS, Institut d Optique Graduate School, Laboratoire Hubert Curien UMR 5516, F-42023, Saint-Etienne, France

April 10, 2019

Abstract

Traditional supervised classification algorithms fail when unlabeled test data arise from a probability distribution that differs from that of the labeled training data. This problem is addressed by domain adaptation, an active research area in which one would like to transfer the knowledge acquired from a first labeled domain, the source, to a second one, the target. In this paper, we tackle this problem from the perspective of large margin classifiers based on $(\epsilon, \gamma, \tau)-$good similarity functions. We first prove a bound on the error of such a classifier on the target domain. Then we present our algorithm consisting in minimizing this bound, allowing to learn a good classifier directly on the target domain without an intermediate domain alignment step. Under specific conditions, our algorithm can be formulated as a convex optimization problem that is solved efficiently. Its performance is assessed via experiments on on a toy set and a real world problem.

Keywords: binary classification, domain adaptation, large margin, similarity functions

1 Introduction

Classification algorithms are used in several real world applications such as image recognition and sentiment analysis. Some of these algorithms output decision rules are based on a pairwise similarity between different data instances, with two of the most known algorithms of this kind being the nearest neighbors classifier [CH67] and the support vector machine [BGV92] (SVM). In order to determine the class of a given example, the former relies on the labels of its nearest neighbors in the sense of a given distance via a voting rule, whereas the latter outputs a classifier that is a linear combination of the point's similarity to the rest of the training data, where the used similarity is restricted to verify Mercer's theorem, i.e to be a kernel. Despite such a restriction, SVM's are appealing due to their great generalization capacity that is empirically observed and theoretically proven [CV95]. In fact, their aim at separating the data with the largest possible margin is a major cause of their success. As a result, it seems interesting to keep this large margin separation aspect without constraining the used similarity function to be a kernel. This is the main topic of the two seminal papers of [BBS08b, BBS08a], which define and analyze the goodness of a similarity function for a given binary classification problem. The definition they introduce is rather intuitive, stating that with a high probability $(1 - \epsilon)$, the average similarity of a data point to landmarks of its own label is greater than that to the opposite label, with the difference between the two average similarities being at least equal to a margin γ. The landmarks are a priori fixed instances that represent a τ fraction of the available data. Given a similarity function verifying this intuitive condition, the authors define a new representation space in which an instance's features are its similarities to the different landmarks, and prove that with enough drawn landmarks, the two classes are linearly separable with a large margin in that representation. Given such encouraging theoretical guarantees, several works in the literature considered the problem of learning such func-

*https://ievred.github.io/
†https://www.creatis.insa-lyon.fr/ lartizien/

tions ([BHS12, GY13, NGHS15, ISH+15]).

The way they are defined, $(\epsilon, \gamma, \tau)-$good similarity functions are convenient for a supervised learning setting in which the training and testing sets arise from the same probability distribution and belong to the same input space. This might not always be the case for real world applications where new partially or totally unlabeled data is available for a given application, but follows a probability distribution that differs from the one generating the labeled training data. An efficient approach to tackle such a problem is transferring the knowledge acquired on the training data to the new test data, and this task is at the heart of the domain adaptation, a currently active research area [PY10, WKW16, Mar11], in which the labeled domain is called the source, while the unlabeled one is called the target. More precisely, domain adaptation offers methodological frameworks and algorithms allowing to leverage information available from both domains to output decision rule that is good for the target one.

In this paper, we consider the aforementioned similarity functions in the challenging setting of unsupervised domain adaptation, where no labels are available for the target domain. We start our contribution by providing a theoretical bound on the error of a given similarity function on the target domain by bounding the error of the corresponding classifier. Up to some terms that are neglected afterwards, the bound is a trade off between a source error term and a disagreement between the two domains. The latter is similar to the $H\Delta H$ divergence [BDBC+10], and its generalization, the discrepancy distance [MMR09]. Following this theoretical contribution, we derive an algorithm that minimizes this bound so that a classifier is directly learnt for the target domain without an additional domain alignment step.

As far as we know, the only works that consider the application of $(\epsilon, \gamma, \tau)-$good similarities in domain adaptation are [MHA12] and [DR18]. The first one uses a heuristic to select landmarks that move closer the two distributions in the projection space, with the similarity function they used being a kernel that is iteratively reweighted. Our work differs from theirs as in our case the similarity function is learnt in one step and using all of the source instances as landmarks. The second establishes theoretical bounds for the error of a similarity function on a target domain in terms of ℓ_1 and χ_2 divergences between probability distributions. In this work, we provide a new bound that involves a domain disagreement term taking into account the considered hypothesis spaces.

In terms of its mechanism consisting in directly learning a classifier on the target domain without an alignment step, our algorithm is similar to the one proposed in [GHLM17]. However, while our domain disagreement term is defined by a supremum, theirs is rather an expectation over the set of considered hypotheses classes, as they consider a PAC-Bayesian setting.

The rest of the paper is organized as follows: the first section introduces required preliminary knowledge and notations. Section 2 is dedicated to our contributions, where we first derive a bound on the error term of the target domain. Then, this bound is used to derive an algorithm by considering a particular case of bilinear similarity functions and linear classifiers, resulting in a a convex programming formulation that is solved efficiently. Finally, in the last section we evaluate our algorithm on a toy data set and on a real world problem.

2 Preliminaries and notations

We consider a binary classification setting, in which the feature space is $\mathcal{X} \subset \mathbb{R}^d$ and the labels set is $\mathcal{Y} = \{-1, 1\}$. Since we work in a domain adaptation context, we suppose having access to a labeled source sample S and an unlabeled target one T, drawn respectively from probability distributions $\mathcal{S}$ and $\mathcal{T}$. Furthermore, we denote by f_S and f_T the two functions labeling the instances of both distributions.

We now recall the definition of a good similarity function $K : \mathcal{X} \times \mathcal{X} \to [-1, 1]$ introduced in [BBS08a]

Definition 1 (Balcan et. al. 2008)**.** *A similarity function K is (ϵ, γ, τ)-good in hinge loss for problem (distribution) $\mathcal{P}$ if there exists a (probabilistic) indicator function R of a set of "reasonable points" such that:*

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{P}} \left[\left(1 - \frac{y.k(x)}{\gamma} \right)_+ \right] \leq \epsilon, \tag{1}$$

$$\mathop{\mathbb{P}}_{x'\sim\mathcal{P}} [R(x')] \geq \tau, \tag{2}$$

where $k(x) = \mathop{\mathbb{E}}_{(x',y')\sim\mathcal{P}} [y' K(x, x') | R(x')]$.

This definition formalizes the intuition that most of instances drawn from a probability distribution $\mathcal{P}$ should have a greater average similarity to landmarks of their own class, than those of the opposite class by a margin γ at least. In fact, $k(x)$ represents the difference between the instance x's average similarity to its own class and to the opposite class, and Equation (1) reflects a penalization for the case where $yk(x) < \gamma$, i.e where the signed difference is not large enough.

Equivalently, k can be seen as a hypothesis used to classify instances, and Equation (1) says that this hypothesis has an expected hinge loss bounded by ϵ at margin γ. We will consider this classifier throughout the next section detailing our contribution.

Given such a similarity function, the authors of [BBS08b, BBS08a] prove that with enough landmark instances, one can construct a new representation space where the features of an instance is its similarities to those landmarks, and where the two classes are linearly separable with a large margin. This result is reminiscent of the kernel trick for SVM's, but it is more general as K does not necessarily have to be a kernel. We note that Definition 1 was modified in [DR18] in a way allowing landmarks to come from a probability distribution that is not necessarily that of the tested data instances.

Using that modified definition, we introduce the following quantity for a probability distribution $\mathcal{P}$ that will be either $\mathcal{S}$ or $\mathcal{T}$ in the rest of the paper.

$$\mathcal{E}_{\mathcal{P}}(k,h) := \mathop{\mathbb{E}}_{x\sim\mathcal{P}}[(1-h(x)k(x)/\gamma_p)_+] \tag{3}$$

where

$$k(x) := \mathop{\mathbb{E}}_{(x',y')\sim\mathcal{S}}[K(x,x')f(x')] \tag{4}$$

γ_p is a margin associated to the probability distribution $\mathcal{P}$ and h is a classifier. We note that regardless of distribution $\mathcal{P}$, landmarks in $\mathcal{E}_{\mathcal{P}}(k,h)$ are drawn from distribution $\mathcal{S}$. In the case where $\mathcal{P}=\mathcal{S}$, this corresponds to Definition 1 with $\tau=1$ and in the case of $\mathcal{P}=\mathcal{T}$, it corresponds to its modification in [DR18].

3 Learning a good classifier for the target domain

We hereby present our contribution, starting by establishing a theoretical bound on the average hinge loss of a classifier k on the target domain. This bound is further used to derive an algorithm that directly learns a good classifier for the target.

3.1 Problem setup

Our goal is to learn a classifier k, or equivalently a similarity function K that has a low error on the target domain $\mathcal{E}_{\mathcal{T}}(k,f_T)$. We follow an approach analogous to the ones presented in [BDBC+10] and [MMR09]. We recall the main result stated in [MMR09, Theorem 8]:

Theorem 1 (Mansour et al., 2009)**.** *Let $\mathcal{H}$ be a hypothesis space, and $L:\mathcal{H}\times\mathcal{H}\to\mathbb{R}^+$ a symmetric loss function verifying the triangle inequality. For a probability distribution $\mathcal{P}$ and $h,g\in\mathcal{H}$, let $\mathcal{L}_{\mathcal{P}}(h,g)=\mathop{\mathbb{E}}_{x\sim\mathcal{P}}[L(g(x),h(x))]$. Let h_S^* and h_T^* as the best classifiers from $\mathcal{H}$ achieving the lowest errors on $\mathcal{S}$ and $\mathcal{T}$ respectively. Finally, define the discrepancy distance between $\mathcal{S}$ and $\mathcal{T}$ as $\mathrm{disc}_L(\mathcal{T},\mathcal{S})=\sup_{h,h'\in\mathcal{H}}|\mathcal{L}_T(h,h')-\mathcal{L}_S(h,h')|$. Then,*

$$\mathcal{L}_T(h,f_T)\le\mathcal{L}_S(h,h_S^*)+\mathrm{disc}(\mathcal{T},\mathcal{S})$$
$$\mathcal{L}_T(h_T^*,f_T)+\mathcal{L}_T(h_T^*,h_S^*)$$

The authors assume that $\mathcal{L}_T(h_T^*,h_S^*)$, the average loss between the best in-class hypotheses for each domain, is small for adaptation to be possible. Furthermore, the term $\mathcal{L}_T(h_T^*,f_T)$ is assumed to be small given that the hypothesis space has enough richness to represent f_T with low error, and $\mathcal{L}_S(h,h_S^*)$ is close to the considered classifier h's error on the source again if f_S is well approximated by h_S^*. This bound will hence be small if h performs well on the source domain and if $\mathcal{S}$ and $\mathcal{T}$ are close in terms of the discrepancy distance.

The above result cannot be used directly as we consider the hinge loss that does not verify the triangular inequality. In order to prove our result, we assume that there exists a function $f:\mathcal{X}\to[-1,1]$ that performs well on the the source and target domains. In the case f achieves a perfect labeling of both the source and target domains, this assumption corresponds in the domain adaptation literature to the covariate shift [SKM07]. Otherwise, it is similar to the ideal joint hypothesis defined in [BDBC+10] and considered in [GHLM17]. While such a function is unknown, we suppose that it belongs to a hypothesis space $\mathcal{H}$ verifying $h\in\mathcal{H}\Rightarrow -h\in\mathcal{H}$.

With these assumptions, we are ready to state our bound on the expected error on the target domain $\mathcal{E}_{\mathcal{T}}(k,f_T)$.

Proposition 1. *Given a similarity function K with a corresponding classifier k, two margins γ_s,γ_t respectively associated to the source and target domains and their ratio $\delta=\frac{\gamma_s}{\gamma_t}$, the following holds:*

$$\mathcal{E}_{\mathcal{T}}(k,f_T)\le\mathcal{E}_{\mathcal{S}}(k,f_S)+\frac{1}{\gamma_s}\sup_{h\in\mathcal{H}}\Delta_\delta(k,h)$$
$$+\mathop{\mathbb{E}}_{x\sim\mathcal{S}}\left[\frac{|f_S(x)-f(x)|}{\gamma_s}\right]+\mathop{\mathbb{E}}_{x\sim\mathcal{T}}\left[\frac{|f_T(x)-f(x)|}{\gamma_t}\right]$$

where $\Delta_\delta(k,h)=\mathop{\mathbb{E}}_{x\sim\mathcal{T}}\left[\left|\mathop{\mathbb{E}}_{x'\sim\mathcal{S}}[k(x')h(x')]-\delta k(x)h(x)\right|\right]$

Proof. We start by writing:

$$\begin{aligned}\mathcal{E}_{\mathcal{T}}(k, f_T) &= \mathcal{E}_{\mathcal{T}}(k, f_T) - \mathcal{E}_{\mathcal{T}}(k, f) && (5)\\ &+ \mathcal{E}_{\mathcal{T}}(k, f) - \mathcal{E}_{\mathcal{S}}(k, f) && (6)\\ &+ \mathcal{E}_{\mathcal{S}}(k, f) - \mathcal{E}_{\mathcal{S}}(k, f_S) && (7)\\ &+ \mathcal{E}_{\mathcal{S}}(k, f_S).\end{aligned}$$

For (7), we have:

$$\begin{aligned}&\mathcal{E}_{\mathcal{S}}(k, f) - \mathcal{E}_{\mathcal{S}}(k, f_S)\\ &= \mathop{\mathbb{E}}_{x\sim\mathcal{S}}\left[(1 - \frac{k(x)f(x)}{\gamma_s})\right] - \mathop{\mathbb{E}}_{x\sim\mathcal{S}}\left[(1 - \frac{k(x)f_S(x)}{\gamma_s})\right]\\ &\leq \frac{1}{\gamma_s}\mathop{\mathbb{E}}_{x\sim\mathcal{S}}\left[k(x)(f_S(x) - f(x))_+\right].\end{aligned}$$

Term (5) can be bounded in the same manner. Concerning term (6), we have:

$$\begin{aligned}&\mathcal{E}_{\mathcal{T}}(k, f) - \mathcal{E}_{\mathcal{S}}(k, f)\\ &= \mathop{\mathbb{E}}_{x\sim\mathcal{T}}\left[\left(1 - \frac{k(x)f(x)}{\gamma_t}\right)_+\right] - \mathop{\mathbb{E}}_{x\sim\mathcal{S}}\left[\left(1 - \frac{k(x)f(x)}{\gamma_s}\right)_+\right]\\ &\leq \mathop{\mathbb{E}}_{x\sim\mathcal{T}}\left[\left(1 - \frac{k(x)f(x)}{\gamma_t}\right)_+\right] - \left(1 - \mathop{\mathbb{E}}_{x\sim\mathcal{S}}\left[\frac{k(x)f(x)}{\gamma_s}\right]\right)_+ && (8)\\ &\leq \mathop{\mathbb{E}}_{x\sim\mathcal{T}}\left[\left(\frac{\mathop{\mathbb{E}}_{x'\sim\mathcal{S}}[k(x')f(x')]}{\gamma_s} - \frac{k(x)f(x)}{\gamma_t}\right)_+\right] && (9)\\ &= \frac{1}{\gamma_s}\mathop{\mathbb{E}}_{x\sim\mathcal{T}}\left[\left(\mathop{\mathbb{E}}_{x'\sim\mathcal{S}}\left[k(x')f(x')\right] - \delta k(x)f(x)\right)_+\right]\\ &\leq \frac{1}{\gamma_s}\sup_{h\in\mathcal{H}}\left(\mathop{\mathbb{E}}_{x\sim\mathcal{T}}\left[\left(\mathop{\mathbb{E}}_{x'\sim\mathcal{S}}\left[k(x')h(x')\right] - \delta k(x)h(x)\right)_+\right]\right)\\ &= \frac{1}{\gamma_s}\sup_{h\in\mathcal{H}}\left(\mathop{\mathbb{E}}_{x\sim\mathcal{T}}\left[\left|\mathop{\mathbb{E}}_{x'\sim\mathcal{S}}\left[k(x')h(x')\right] - \delta k(x)h(x)\right|\right]\right)\end{aligned}$$

where (8) is obtained by applying Jensen's inequality to the convex function $(1 - \cdot)_+$. Then, using the inequality $(t)_+ - (s)_+ \leq (t - s)_+$ (sub-additivity of the positive part), one gets line (9). The last line is a consequence of the fact that $h \in \mathcal{H} \Rightarrow -h \in \mathcal{H}$. □

The established bound has 4 terms: the first one is the error on the source w.r.t function f_S, known through the labels y. This term is similar to $\mathcal{L}_S(h, h_S^*)$ in theorem 1.

The second term reflects a disagreement between the source and the target w.r.t hypothesis k. Its counterpart in 1 is the discrepancy disc, and if $\mathcal{K} = \mathcal{H}$, both disagreement measures bound the same quantity:

$$\begin{aligned}\mathcal{E}_{\mathcal{T}}(k, f) - \mathcal{E}_{\mathcal{S}}(k, f) &\leq \sup_{h,h'\in\mathcal{H}}|\mathcal{E}_{\mathcal{T}}(h, h') - \mathcal{E}_{\mathcal{S}}(h, h')|\\ &= \mathrm{disc}(\mathcal{T}, \mathcal{S})\end{aligned}$$

$$\mathcal{E}_{\mathcal{T}}(k, f) - \mathcal{E}_{\mathcal{S}}(k, f) \leq \frac{1}{\gamma_s}\sup_{x\in}\Delta_\delta(k, h)$$

The last two terms are distances between f and the best hypotheses in $\mathcal{H}$ that respectively label elements drawn from $\mathcal{S}$ and $\mathcal{T}$. We will neglect both of them in our algorithm's definition, as we would expect them to be small for the adaptation to be possible, given the existence of f. After omitting these two terms, the sum of the two remaining terms defines our algorithm that we detail in the next section.

3.2 Algorithm derivation

We suppose searching for the similarity function K in hypothesis space, and we denote by $\mathcal{K}$ the hypothesis space of classifiers it induces, i.e from which the resulting classifier k is picked. Our algorithm is then formulated as follows:

$$\underset{k\in\mathcal{K}}{\text{minimize}}\ \mathcal{E}_{\mathcal{S}}(k, f_S) + \frac{1}{\gamma_s}\sup_{h\in\mathcal{H}}\Delta_\delta(k, h)$$

The cost function in this case contains a supremum over the potentially infinite hypothesis set $\mathcal{H}$, which makes the optimization difficult. However, we will show in the next section that a particular choice of $\mathcal{H}$ allows to deal efficiently with this term, transforming it into a maximum over a finite set.

Using the Lagrange multipliers, the minimization problem is equivalent to:

$$\begin{aligned}&\underset{k\in\mathcal{K}}{\text{minimize}}\ \sup_{h\in\mathcal{H}}\Delta_\delta(k, h)\\ &\text{subject to } \mathcal{E}_{\mathcal{S}}(k, f_S) \leq \epsilon\end{aligned}$$

where ϵ is a hyperparameter. Without the constraint on the source domain error, the trivial null similarity function $K \equiv 0$ is a solution to the problem. Thus, it plays a major role in avoiding this solution, in addition to imposing a low error on the source domain.

We note that if the constrained version of the minimization algorithm manages to find a small value for $\sup_{h\in\mathcal{H}}\Delta_\delta(k, h)$, then from the proof of Proposition 1, we have

$$\begin{aligned}\mathop{\mathbb{E}}_{x\sim\mathcal{T}}\left[\left(\mathop{\mathbb{E}}_{x'\sim\mathcal{S}}[k(x')f(x')] - \delta k(x)f(x)\right)_+\right] &= \Delta_\delta(k, f)\\ &\leq \sup_{h\in\mathcal{H}}\Delta_\delta(k, h).\end{aligned}$$

The left hand side of these inequalities penalizes the case $\frac{1}{\delta}\mathop{\mathbb{E}}_{x'\sim\mathcal{S}}[k(x')f(x')] > k(x)f(x)$ for every instance

x drawn from the target distribution, resulting in similarity K that is good on $\mathcal{T}$ with margin $\bar{\gamma} = \frac{\gamma_t}{\gamma_s} \mathop{\mathbb{E}}_{x\sim\mathcal{S}} \left[\mathop{\mathbb{E}}_{x'\sim\mathcal{S}} [K(x,x')f(x)f_S(x')] \right]$. This is the mean margin of K on the source domain when f and f_S label respectively the data points and the landmarks, up to a scaling factor $\frac{\gamma_t}{\gamma_s}$. This also means that the classifier k would have a low error on the target with a margin $\frac{\bar{\gamma}}{\gamma_s}\gamma_t$, which is a scaled version of the originally considered margin γ_t for the target domain.

3.3 Case of bilinear similarities and linear classifiers

Below, we discuss a particular choice for both hypotheses classes $\mathcal{K}$ and $\mathcal{H}$ in order to make the optimization problem tractable. To this end, we consider bilinear similarity functions $K(x,x') = x^T A x'$ with $A \in \mathbb{R}^{d\times d}$, $\|A\| \leq 1$. Such functions were studied in the $(\epsilon,\gamma,\tau)-$good framework in [BHS12]. By scaling the data instances so that their Euclidean norms are bounded by 1, K takes values in $[-1,1]$. We proceed to determine the implied $\mathcal{K}$ space in this case. Let $x \in \mathbb{R}^d$:

$$\begin{aligned} k(x) &= \mathop{\mathbb{E}}_{x'\sim\mathcal{S}} [K(x,x')f_S(x')] \\ &= \mathop{\mathbb{E}}_{x'\sim\mathcal{S}} \left[x^T A x' f_S(x')\right] \\ &= x^T A \mu \end{aligned}$$

with $\mu = \mathop{\mathbb{E}}_{x\sim\mathcal{S}} [x.f_S(x)]$ and

$$\begin{aligned} \mathcal{K} =& \{k : x \mapsto x^T A\mu; A \in \mathbb{R}^{d\times d} \|A\| \leq 1\} \\ \simeq& \{A\mu; A \in \mathbb{R}^{d\times d}; \|A\| \leq 1\} \\ \subset& \{a \in \mathbb{R}^d; \|a\| \leq 1\}, \end{aligned}$$

where the $\simeq$ symbol denotes equality up to an isomorphism of vector spaces.

As for $\mathcal{H}$, we choose the space of linear classifiers with bounded ℓ_1 norm:

$$\mathcal{H} \simeq \{w \in \mathbb{R}^d; \|w\|_1 \leq 1\}.$$

Hence, for $k(x) = a^T x$ and $h(x) = w^T x$, one has

$$\begin{aligned} \Delta_\delta(a,w) &= \mathop{\mathbb{E}}_{x\sim\mathcal{T}} \left[\left(\mathop{\mathbb{E}}_{x'\sim\mathcal{S}} \left[a^T x' x'^T w\right] - \delta a^T x x^T w \right)_+ \right] \\ &= \mathop{\mathbb{E}}_{x\sim\mathcal{T}} \left[\left| a^T (\mathop{\mathbb{E}}_{x'\sim\mathcal{S}} \left[x' x'^T\right] - \delta x x^T) w \right| \right]. \end{aligned}$$

For a fixed $a \in \mathbb{R}^d$, the function $w \mapsto \Delta_\delta(a,w)$ is convex, hence its supremum over $\mathcal{H}$, an ℓ_1 unit ball, is reached on one of its vertices

$$\sup_{\|w\|_1\leq 1} \Delta_\delta(a,w) = \max_{1\leq i\leq d} \left(\mathop{\mathbb{E}}_{x\sim\mathcal{T}} \left[\left| a^T (\Sigma_S - \delta x x^T) e_i \right| \right] \right),$$

where $\Sigma_S = \mathop{\mathbb{E}}_{x\sim\mathcal{S}} \left[x x^T\right]$ and $\{e_1, ..., e_d\}$ is the canonical basis of $\mathbb{R}^d$. Multiplying the cost function by γ_s, the problem is written:

$$\begin{aligned} &\underset{\|a\|\leq 1}{\text{minimize}} \mathop{\mathbb{E}}_{x\sim\mathcal{S}} [(\gamma_s - f_S(x)k(x))_+] + M \\ &\text{s.t. } M \geq \mathop{\mathbb{E}}_{x\sim\mathcal{T}} \left[\left| a^T (\Sigma_S - \delta x x^T) e_i \right| \right] \forall 1 \leq i \leq d. \end{aligned}$$

In the empirical case, i.e when the expectation terms are replaced by corresponding empirical means over the source and target samples of respective sizes m and n, this is a quadratic optimization problem having $d+m$ variables (d for the size of vector a, m for the positive variables representing the positive parts) and $1+2m+2nd$ constraints (one for $\|a\|$, 2 for each source instance and $2d$ for each target one). It can be solved efficiently using standard convex optimization solvers.

4 Experiments

In this section, we evaluate our method on two domain adaptation problems. The first is defined by a toy set with a controllable difficulty, while the second is a real world problem.

4.1 Cross-validation

We choose the best values of hyperparameters γ_s and δ by a reverse validation procedure ([ZFY+10], [BM10]) following the protocol of [GHLM17] with a 5 folds validation. Given a fold $i \in 1, ..., 5$ defining a training set $S \setminus S_i$ and a validation set and S_i for the source, and similarly $T \setminus T_i$ and T_i for the target, a classifier h is learnt from labeled $S \setminus S_i$ and unlabeled $T \setminus T_i$. Then, keeping the same hyperparameters used to learn h, a reverse classifier h_r is learnt from $T \setminus T_i$ labeled by h and unlabeled $S \setminus S_i$, and finally evaluated on S_i. The chosen hyperparameters are those minimizing the average error of h_r over the folds. This way we do not make use of the target labels, which fits with the unsupervised domain adaptation setting. We slightly modify this procedure to suit our method: γ_s and δ, or equivalently γ_s and γ_t are respectively associated with $\mathcal{S}$ and $\mathcal{T}$, hence when computing the reverse classifier h_r, we replace γ and δ respectively by $\frac{\gamma_s}{\delta} = \gamma_t$ and $\frac{1}{\delta} = \frac{\gamma_t}{\gamma_s}$. Moreover, the best hyperparameters are those minimizing the average margin violation loss of h_r on the source validation sets S_i, i.e

$$\frac{1}{5} \sum_{i=1}^{5} \left(\frac{1}{|S_i|} \sum_{x\in S_i} [y(x)h_r(x) < \gamma_s] \right)$$

Angle (°)	20	30	40	50	70	90
SVM (no adaptation) [CFTR15]	10.4	24	31.2	40	76.4	82.8
DASVM [BM10]	**0**	25.9	28.4	33.4	74.7	82
PBDA [GHLM17]	9.4	10.3	22.5	41.2	62.6	68.7
OT-GL [CFTR15]	**0**	**0**	**1.3**	19.6	**37.8**	**50.8**
DASF [MHA12]	**0.20**	**0.45**	8.97	18.73	38.05	**40.25**
Our algorithm (8 KPCA features)	1.31	2.37	**3.56**	**12.9**	54.45	72.41
Our algorithm (20 KPCA features)	0.84	4.11	8.76	**13.93**	**36.19**	56.32

Table 1: Average 0-1 loss (percentage) over 10 realizations for the moons toy set.

where [...] denotes the Iverson bracket. After choosing the hyperparameters by this procedure, the resulting classifier's performance is evaluated on two independent source and target sets. This whole procedure (reverse validation then testing) is repeated 10 times, and the average performances over those repetitions are reported. For all of the experiments, we use the `CVXPY` modeling language ([DB16], [AVDB18]) with the solver `MOSEK` ([ApS17]).

4.2 Moons data set

We carry on our experiments on the moons data set used in [GHLM13] and [CFTR15]. The source data set is centered at the origin $(0, 0)$, and has 300 instances, which are rotated around that point by a certain angle to get the target distribution. Obviously the greater is the angle, the further from each other are the two domains and the harder is the adaptation. To ensure the data is linearly separable, we apply a Kernel PCA [SSM97] with a Gaussian kernel having a parameter σ equal to the mean Euclidean distance between instances (as done in [KJ11]), keeping respectively 8 and 20 components. The mean over 10 tests on independent data sets of 1000 instances are reported in Table 1, where our algorithm is compared to an SVM trained on the source domain (without adaptation) and domain adaptation algorithms DASVM [BM10], PBDA [GHLM17], OT-GL [CFTR15] and DASF[MHA12]. We observe that our method outperforms both DASVM and PBDA for angles up to 70°. However, its performance remains lower than DASF and OT-GL except for angles 50°and 70°. We think that this difference is due to the fact that our bound overestimates the divergence between the two domains for small angles due to the supremum term in Proposition 1.

4.3 Prostate cancer data set

We test our algorithm on a real world problem: a clinical data set of multi-parametric magnetic resonance images (mp-MRI) collected to train a computer-aided diagnosis system for prostate cancer mapping [NRML$^+$12, ALP$^+$15]. This system learns a binary decision model in a multidimensional feature space based on training samples (voxels) from different classes of interest. This model is then used to generate cancer probability maps.

Data description The considered source and target data are mp-MRI exams of 90 patients acquired on two different MRI scanners (1.5 T and 3T) and thus leading to images with different resolution and texture patterns. Details are given in Table 2.

Scanner (domain)	1.5T	3T
Number of patients	49	41
Number of voxels	419348	987396
% of positive class	13.38	14.26

Table 2: mp-MRI data description.

Each individual voxel is described by a binary label (Cancer, Non Cancer) and a set of 95 handcrafted features consisting of image descriptors, texture coefficients, gradients and other visual characteristics (more details in [NRML$^+$12]).

To tackle the class imbalance illustrated in Table 2, we randomly select a balanced 2000 voxels data set for each of the 10 repeated reverse validation procedures. Moreover, we run a principal component analysis keeping 95% of the total variance, and reducing the number of features to 28. With the best found hyperparameters, our algorithm is evaluated 10 times on independent data sets of 10000 instances for each domain. Results are reported in Table 3, where we compare our algorithm with a linear SVM without adaptation

(`Scikit-Learn` [PVG+11]'s implementation). We notice that for both domain adaptation where we consider the 3T and 1.5T domains as source and target interchangeably, the scores on the target domain are close, reflecting a symmetry of the problem. Our method exhibits an improvement of 11% over the case without adaptation.

Problem	3T→1.5T	1.5T→3T
Linear SVM (no adaptation)	49.99	49.56
Our algorithm	38.99	38.39

Table 3: Average 0-1 loss (percentage) over 10 realizations for the mp-MRI data set.

5 Conclusion and future perspectives

In this paper, we presented a new algorithm for domain adaptation that is suitable for large margin classifiers. Our algorithm is derived from a theoretical bound and allows to learn a classifier on the target domain directly without an additional alignment step. In the case of bilinear similarity functions and linear classifiers, it is formulated as a quadratic optimization problem that is solved efficiently. The results we obtain are encouraging, although not surpassing state of the art methods ([CFTR15], [MHA12]), and suggest trying to obtain a tighter bound from which the algorithm is derived or using a domain disagreement term that is less strict than a supremum. For example, considering an expectation over the considered set of classifiers [GHLM17] is worth a try. Moreover, in the context of large margin classifiers, questions about our method's generalization guarantees are naturally raised. Finally, the multi-class case and the semi-supervised domain adaptation (i.e when some labels are available on the target) are crucial extensions to add as they would allow us to test on more real world data sets.

References

[ALP+15] Rahaf Aljundi, Jérôme Lehaire, Fabrice Prost-Boucle, Olivier Rouvière, and Carole Lartizien. Transfer learning for prostate cancer mapping based on multicentric MR imaging databases. In *Machine Learning Meets Medical Imaging workshop at ICML*, pages 74–82, 2015.

[ApS17] MOSEK ApS. *MOSEK Optimizer API for Python 8.1.0.80*, 2017.

[AVDB18] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[BBS08a] Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. *Improved guarantees for learning via similarity functions.* 2008.

[BBS08b] Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. A theory of learning with similarity functions. *Machine Learning*, 72(1), 2008.

[BDBC+10] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1), 2010.

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 1992.

[BHS12] Aurélien Bellet, Amaury Habrard, and Marc Sebban. Similarity Learning for Provably Accurate Sparse Linear Classification. In *ICML*, 2012.

[BM10] L. Bruzzone and M. Marconcini. Domain Adaptation Problems: A DASVM Classification Technique and a Circular Validation Strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5), 2010.

[CFTR15] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal Transport for Domain Adaptation. *arXiv:1507.00504 [cs]*, 2015. arXiv: 1507.00504.

[CH67] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 1967.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3), 1995.

[DB16] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[DR18] Sofiane Dhouib and Ievgen Redko. Revisiting (ϵ, γ, τ)-similarity learning for domain adaptation. In *Advances in Neural Information Processing Systems 31*. 2018.

[GHLM13] Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. A PAC-Bayesian Approach for Domain Adaptation with Specialization to Linear Classifiers. In *International Conference on Machine Learning*, 2013.

[GHLM17] Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. PAC-Bayes and Domain Adaptation. *arXiv:1707.05712 [stat]*, 2017. arXiv: 1707.05712.

[GY13] Zheng-Chu Guo and Yiming Ying. Guaranteed Classification via Regularized Similarity Learning. *arXiv:1306.3108 [cs]*, 2013. arXiv: 1306.3108.

[ISH+15] Nicolae Irina, Marc Sebban, Amaury Habrard, Eric Gaussier, and Massih-Reza Amini. Algorithmic Robustness for Semi-Supervised (ϵ, γ, τ)-Good Metric Learning. In *ICONIP*, 2015.

[KJ11] Purushottam Kar and Prateek Jain. Similarity-based Learning via Data Driven Embeddings. In *Advances in Neural Information Processing Systems 24*. 2011.

[Mar11] Anna Margolis. A Literature Review of Domain Adaptation with Unlabeled Data. 2011.

[MHA12] Emilie Morvant, Amaury Habrard, and Stéphane Ayache. Parsimonious unsupervised and semi-supervised domain adaptation with good similarity functions. *Knowledge and Information Systems*, 33(2), 2012.

[MMR09] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain Adaptation: Learning Bounds and Algorithms. *arXiv:0902.3430 [cs]*, 2009. arXiv: 0902.3430.

[NGHS15] Maria-Irina Nicolae, Éric Gaussier, Amaury Habrard, and Marc Sebban. Joint Semi-supervised Similarity Learning for Linear Classification. In *ECML/PKDD*, volume 9284. 2015.

[NRML+12] Emilie Niaf, Olivier Rouvière, Florence Mège-Lechevallier, Flavie Bratan, and Carole Lartizien. Computer-aided diagnosis of prostate cancer in the peripheral zone using multiparametric mri. *Physics in medicine and biology*, 57(12):3833–51, 2012.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[PY10] S. J. Pan and Q. Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), 2010.

[SKM07] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate Shift Adaptation by Importance Weighted Cross Validation. *Journal of Machine Learning Research*, 8(May), 2007.

[SSM97] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Artificial Neural Networks — ICANN'97*, Lecture Notes in Computer Science, 1997.

[WKW16] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1), 2016.

[ZFY+10] Erheng Zhong, Wei Fan, Qiang Yang, Olivier Verscheure, and Jiangtao Ren. Cross Validation Framework to Choose amongst Models and Datasets for Transfer Learning. In *ECML/PKDD*, volume 6323. 2010.

Apprentissage automatique sur des données de type graphe utilisant le plongement de Poincaré et les algorithmes stochastiques riemanniens

Hatem Hajri, Hadi Zaatiti, Georges Hebrail, et Patrice Aknin

Institut de recherche technologique SystemX, 8 Avenue de la Vauve, 91120 Palaiseau, France.

Email: `firstname.name@irt-systemx.fr`

5 juin 2019

Résumé

Afin de mieux capter la complexité des relations qui existent entre les nœuds d'un graphe binaire, des travaux originaux ont montré l'intérêt de représenter ces données dans des espaces hyperboliques.

Ces travaux sont issus d'une première communauté qui s'intéresse aux données graphiques et à leurs représentations et visualisations. D'autre part, une seconde communauté qui adresse plutôt des applications en traitement des données Radar et en vision par ordinateur, a développé ces dernières années des outils d'apprentissage sur certains espaces hyperboliques en exploitant leurs intéressantes propriétés géométriques.

Dans cet article, nous présentons nos travaux récents [13, 27] qui visent à rapprocher ces deux approches. Plus précisément, nous combinons les plongements des graphes et les méthodes récentes de partitionnement sur les espaces hyperboliques dans le but de réaliser une classification par apprentissage sur les données initiales du graphe. Nous illustrons cette proposition par des applications en montrant le gain obtenu vis-à-vis de l'état de l'art.

Mots-clef : Plongement de Poincaré, Barycentre Riemannien, Algorithme des K-moyennes Riemannien, Algorithme espérance-maximisation Riemannien.

1 Introduction

L'idée de plonger des données dans un nouvel espace pour en faciliter le traitement a démontré son efficacité dans de multiples applications. Les techniques comme Word2vec [15], Glove [18], Node2vec [12], Graph2vec [16] et DeepWalk [19] sont devenues célèbres grâce à leurs capacité à représenter les données tout en réduisant la complexité de l'espace initial pour les traitements ultérieurs. Des travaux récents comme [17, 10, 23, 24] montrent que les représentations des graphes dans des espaces hyperboliques préservent plus fidèlement la structure sous-jacente ainsi que les relations latentes entre les noeuds d'un graphe que leurs homologues euclidiens.

Dans le but de partitionner les noeuds d'un graphe binaire donné, plusieurs équipes proposent de plonger d'abord le graphe dans un espace euclidien avec les techniques citées (Node2vec, Graph2vec, DeepWalk) et ensuite d'appliquer des méthodes classiques de partitionnement comme les algorithmes des K-moyennes et d'espérance-maximisation (EM) [28, 25, 26, 9, 28]. Nos articles récents [13, 27] proposent des versions hyperboliques de ces travaux, motivés en grande partie par les bons résultats obtenus en traitement de données Radar [3] et en vision par ordinateur [7, 21, 22]. En particulier [7] utilise le barycentre riemannien sur l'espace des matrices de covariances pour classifier des signaux EEG. [3] utilise la médiane riemannienne sur le disque de Poincaré pour détecter les données aberrantes parmi des signaux Radar. [21] introduit l'algorithme EM sur l'espace des matrices de covariance et l'applique au problème de classification des images. Ces applications se basent sur les propriétés intéressantes des espaces hyperboliques qui seront rappelées dans ce papier.

L'article est organisé comme suit. La section 2 présente un rappel sur la géométrie riemannienne du disque de Poincaré comme un modèle typique de géométrie hyperbolique et introduit la version rieman-

nienne des algorithmes des K-moyennes et d'espérance-maximisation. La section 3 rappelle la méthode récente du plongement des données de type graphe dans le disque de Poincaré [17]. Nous complétons ensuite cette section par deux propositions pour l'apprentissage supervisé et non supervisé sur les graphes. La section 4 présente des expérimentations démontrant l'intérêt des approches proposées tout en les comparant avec l'état de l'art récent. Enfin la section 5 conclut l'article en esquissant les perspectives de ce travail.

2 Apprentissage statistique riemannien sur le disque de Poincaré

Dans cette section, nous introduisons les outils mathématiques nécessaires qui seront utilisés pour la partie apprentissage automatique. Nous commençons par rappeler la géométrie du disque de Poincaré. Ensuite, nous présentons deux extensions des algorithmes K-moyennes et espérance maximisation sur cet espace.

Le disque de Poincaré est l'espace noté $\mathbb{D} = \{z \in \mathbb{C} : |z| < 1\}$, muni de la distance dite de Poincaré $d(z_0, z_1)$ définie par :

$$\frac{1}{2}\log\left(\left(1+\left|\frac{z_1-z_0}{1-\bar{z_0}z_1}\right|\right)\left(1-\left|\frac{z_1-z_0}{1-\bar{z_0}z_1}\right|\right)^{-1}\right)$$

Par ailleurs, rappelons qu'une transformation de Möbius est une application de la forme $f_{a,b} : z \in \mathbb{D} \mapsto \frac{az+b}{\bar{b}z+\bar{a}}$ où $a, b \in \mathbb{C}$ et $|a|^2 - |b|^2 = 1$. $\mathcal{M}$, l'ensemble de toutes les transformations de Möbius est un groupe pour la composition qui agit sur $\mathbb{D}$ transitivement (i) et isométriquement (ii) :

(i) pour tous $z_1, z_2 \in \mathbb{D}$, il existe $f_{a,b} \in \mathcal{M}$ t.q $f_{a,b}(z_1) = z_2$

(ii) pour tous $z_1, z_2 \in \mathbb{D}$, $d(f_{a,b}(z_1), f_{a,b}(z_2)) = d(z_1, z_2)$

Muni du groupe $\mathcal{M}$, $\mathbb{D}$ devient un espace riemannien symétrique homogène de courbure négative [14].

Une propriété importante satisfaite par ces espaces est l'existence et l'unicité du barycentre riemannien (aussi appelé moyenne riemannienne ou moyenne de Fréchet). Plus précisément, pour tout ensemble $\{z_i, 1 \leq i \leq n\}$ inclus dans $\mathbb{D}$, la solution du problème d'optimisation suivant (barycentre riemannien)

$$\hat{z}_n = \text{argmin}_{z\in\mathbb{D}}\left(\sum_{i=1}^{n} d^2(z, z_i)\right) \tag{1}$$

existe, est unique et appartient à $\mathbb{D}$ [2]. $\hat{z}_n$ peut être approché numériquement par une extension au cadre riemannien de la méthode de descente de gradient classique. Ceci a fait l'objet de plusieurs travaux comme [8, 4, 6, 5, 3]. Dans la suite, nous allons utiliser l'algorithme (1) proposé dans [3]. Les notations $\exp_z$ et $\log_z$ désignent respectivement les fonctions exponentielle et logarithme riemanniennes.

Algorithm 1 Calcul approché du barycentre riemannien sur $\mathbb{D}$ par descente de gradient

Entrée : $Z = \{z_i, 1 \leq i \leq n\}$: sous ensemble de $\mathbb{D}$, z_{init} : initialisation du barycentre, τ : pas de la descente de gradient, ε : précision d'arrêt

Sortie : $\hat{z}$: approximation du barycentre riemannien

1: $\hat{z} \leftarrow z_{init}$

2: $d = 1e^6$ ▷ un nombre assez large

3: **while** $d > \varepsilon$ **do**

4:

$$\mu \leftarrow \frac{2}{n}\sum_{i=1}^{n}\log_{\hat{z}}(z_i), \hat{z} \leftarrow \exp_{\hat{z}}(\tau\mu), d \leftarrow \sqrt{\frac{|\mu|^2}{(1-|\hat{z}|^2)^2}}$$

5: **end while** **return** $\hat{z}$

Dans les deux paragraphes suivants, nous présentons les algorithmes des K-moyennes et espérance-maximisation riemanniens qui exploitent les propriétés intéressantes de l'espace de Poincaré.

2.1 Algorithme des K-moyennes riemannien

Par analogie avec le cas euclidien, nous pouvons étendre l'algorithme des K-moyennes au cadre riemannien. Cette extension est basée sur l'observation que pour un ensemble $\{x_1, \cdots, x_n\}$, la moyenne $\hat{x}_n = \frac{1}{n}\sum_i x_i$ correspond aussi au minimum global de la fonction $\sum_{i=1}^{n}(x - x_i)^2$. Dans la version riemannienne, la distance euclidienne sera remplacée par son homologue riemannien.

2.2 Algorithme EM riemannien

En exploitant les propriétés de la distance de Poincaré, il est possible d'étendre l'algorithme EM classique à cet espace [22]. Cette extension s'appuie sur une version riemannienne de la loi gaussienne.

Étant donné deux paramètres $\bar{z} \in \mathbb{D}$ (une position) et $\sigma > 0$ (un écart type), la loi gaussienne $G(\bar{z}, \sigma)$ dé-

Algorithm 2 K-moyennes riemannien sur $\mathbb{D}$

Entrée : K : nombre de classes, Z : un ensemble de n complexes appartenant à $\mathbb{D}$

Sortie : N : ensemble de K barycentres, *labels* : n étiquettes des données d'entrée Z

1: **Initialisons** K barycentres, $N = \{\nu_1, \nu_2, ..., \nu_K\}$ aléatoirement dans $\mathbb{D}$
2: **repeat**
3: **for** $z_i \in Z$ **do** $c_i \leftarrow argmin_{j\in\{1,...,K\}} d(z_i, \nu_j)$
4: **for** $j \in \{1, ..., K\}$ **do**
5: $\nu_j \leftarrow$ `Barycentre_Riemannien`$(\{z_i | c_i = j\})$
6: **end for**
7: **until** les barycentres deviennent stables
8: **return** N, $labels = \{c_i\}$

pendant de $(\bar{z}, \sigma)$ est définie dans [22] par sa densité

$$p(z|\bar{z}, \sigma) = \frac{1}{\zeta(\sigma)} \exp\left[-\frac{d^2(z, \bar{z})}{2\sigma^2}\right]$$

relativement au volume riemannien $dv(z)$. Une propriété clé de cette distribution, découlant de la symétrie de $\mathbb{D}$, est que $\zeta(\sigma)$ ne dépend pas de $\bar{z}$ (tout comme la loi gaussienne classique). Ceci permet d'en déduire les estimateurs du maximum de vraisemblance (EMV) de $\bar{z}$ et σ étant donné un échantillon $z_1, \cdots, z_N$ de $G(\bar{z}, \sigma)$ comme suit.

EMV de $\bar{z}$. L'EMV de $\bar{z}$ noté $\hat{z}_N$ est le barycentre riemannien empirique de $z_1, \cdots, z_N$.

EMV de σ. L'EMV de σ est $\hat{\sigma}_N = \Phi(\frac{1}{N}\sum_{n=1}^{N} d^2(\hat{z}_N, z_n))$ où Φ est l'inverse de la fonction $\sigma \mapsto \sigma^3 \times \frac{d}{d\sigma} \log \zeta(\sigma)$

En pratique, comme $\hat{z}_N$, $\hat{\sigma}_N$ n'a pas d'expression explicite mais peut être approché numériquement par des méthodes de Newton. Considérons maintenant un modèle de mélange gaussien

$$p(z|(\varpi_\mu, \bar{z}_\mu, \sigma_\mu)_{1\leq\mu\leq M}) = \sum_{\mu=1}^{M} \varpi_\mu \times p(z|\, \bar{z}_\mu, \sigma_\mu)$$

où les ϖ_μ sont positifs de somme 1. Les paramètres du modèle $\varpi_\mu, \bar{z}_\mu, \sigma_\mu$ peuvent être estimés par une extension de l'algorithme EM classique [22]. Pour ceci, introduisons pour tout $\vartheta = \{(\varpi_\mu, \bar{z}_\mu, \sigma_\mu)\}$,

$$\omega_\mu(z_n, \vartheta) = \frac{\varpi_\mu \times p(z_n|\, \bar{z}_\mu, \sigma_\mu)}{\sum_{s=1}^{M} \varpi_s \times p(z_n|\, \bar{z}_s, \sigma_s)}$$

et

$$N_\mu(\vartheta) = \sum_{n=1}^{N} \omega_\mu(z_n)$$

L'algorithme EM riemannien met à jour $\hat{\vartheta} = \{(\hat{\varpi}_\mu, \hat{Y}_\mu, \hat{\sigma}_\mu)\}$ de façon itérative pour générer l'EMV de $\vartheta = (\varpi_\mu, \bar{Y}_\mu, \sigma_\mu)$ de la manière suivante.

► Mise à jour de $\hat{\varpi}_\mu$: $\hat{\varpi}_\mu = N_\mu(\hat{\vartheta}) \big/ N$.

► Mise à jour de $\hat{z}_\mu$ (en utilisant une descente de gradient riemannienne) :

$$\hat{z}_\mu = \text{argmin}_z \sum_{n=1}^{N} \omega_\mu(z_n, \hat{\vartheta})\, d^2(z, z_n)$$

► Mise à jour de $\hat{\sigma}_\mu$ (en utilisant une méthode de Newton) :

$$\hat{\sigma}_\mu = \Phi(\, N_\mu^{-1}(\hat{\vartheta}) \times \textstyle\sum_{n=1}^{N} \omega_\mu(z_n, \hat{\vartheta})\, d^2(\hat{z}_\mu, z_n))$$

3 Apprentissage sur les graphes en utilisant le plongement de Poincaré

Dans cette section, nous présentons une approche récente [17] qui propose le plongement d'un graphe binaire dans un disque de Poincaré. Ensuite, nous proposons deux algorithmes de partitionnement des noeuds d'un graphe dans les cas supervisé et non supervisé.

Considérons un graphe $(\mathcal{V}, \mathcal{E})$ où $\mathcal{V}$ est l'ensemble des noeuds et $\mathcal{E}$ est l'ensemble des arrêtes. Pour plonger $\mathcal{V}$ dans $\mathbb{D}$, [17] propose d'apprendre une application $u \in \mathcal{V} \mapsto \theta_u \in \mathbb{D}$ minimisant la fonction

$$\mathcal{L}(\Theta) = \log(\sigma(-d(\theta_u, \theta_v))) + \sum_{i=1}^{M} \mathbb{E}_{P_\mathcal{N}}[\log(\sigma(d(\theta_{u_i}, \theta_v)))]$$

où $u_1, \cdots, u_M$ est un ensemble d'échantillons négatifs tirés suivant une distribution $P_\mathcal{N}$ et $\sigma(x) = (1+e^{-x})^{-1}$ est la fonction softmax [15]. En minimisant cette fonction, θ_u et θ_v se rapprochent l'un de l'autre et les noeuds des échantillons négatifs s'éloignent de θ_v. En pratique $\mathcal{L}(\Theta)$ est optimisé en générant des noeuds sur le graphe à l'aide de l'algorithme DeepWalk [19] puis en tirant des échantillons négatifs suivant la distribution unigram à la puissance 3/4 [15].

Étant donne un cluster $\mathcal{C} = \{z_i, 1 \leq i \leq n\}$ dans $\mathbb{D}$, ayant comme barycentre $\hat{z}$, on définit sa variance empirique par

$$\text{Var}(\mathcal{C}) = \frac{1}{n} \sum_{i=1}^{n} d^2(\hat{z}, z_i)$$

L'algorithme 3 ci-dessous présente notre schéma pour effectuer un partitionnement non supervisé sur un

graphe. L'idée est de générer plusieurs plongements de Poincaré et de garder le plongement le plus concentré après déroulement d'un algorithme K-moyennes. Pour notre travail, nous allons considérer comme meilleur, le plongement ayant la variance minimale (d'autres métriques sont possibles). Dans ce qui suit, la fonction `Poincare_Plongement` plonge un graphe sur le disque de Poincaré en minimisant $\mathcal{L}(\Theta)$.

Algorithm 3 Apprentissage non supervisé

Entrée : $\mathcal{G}$: graphe binaire, K : nombre de clusters (entier), NE : nombre d'expérimentations

Sortie : $Meilleur\_Plongement$: Plongement des noeuds du graphe d'entrée ayant la variance totale minimale, N : barycentres des K clusters, $labels$: étiquette de chaque noeud

1: **repeat**

2: $$Plongement_i \leftarrow \texttt{Poincare\_Plongement}(\mathcal{G})$$

3: $$N_i, labels_i \leftarrow \texttt{K\_moyennes}(K, Plongement_i)$$

4: $$Clusters \leftarrow \{c_1, ..., c_K\}$$

5: **tel que** $c_j = \{z_q \in Plongement_i | labels_i[q] = j\}$

6: $$var_i = max_{j \in \{1,...,K\}}(Var(c_j))$$

7: **until** NE plongements sont effectuées

8: $$Meilleur\_Plongement \leftarrow Plongement_{meilleur}$$

9: **tel que** $meilleur = argmin_{i \in \{1,...,NE\}}(var_i)$

10: **return**

11: $Meilleur\_Plongement, N_{Meilleur}, labels_{Meilleur}$

Cet algorithme peut être suivi aussi par un algorithme EM riemannien pour obtenir une classification plus robuste tenant compte de la dispersion des points au sein des clusters.

L'algorithme 4 ci-dessous présente le schéma proposé pour l'apprentissage supervisé. Bien que cet algorithme puisse être alimenté par l'algorithme 3, une version simple et indépendante de celui ci est présentée ici.

La seconde alternative proposée réalise le plongement de la vérité de terrain selon un mélange gaussien en utilisant l'algorithme EM dans un premier temps et

Algorithm 4 Apprentissage supervisé

Entrée : $\mathcal{G}$: graphe binaire, VT : une vérité terrain pour chaque noeud de $\mathcal{G}$

Sortie : $Cluster$: cluster d'appartenance de chaque noeud des données d'apprentissage

1: $Plongement \leftarrow \texttt{Poincare\_Plongement}(\mathcal{G})$

2: $$DE, DT \leftarrow \texttt{Diviser}(Plongement)$$

▷ DE sont les données d'entraînements

▷ DT sont les données de tests

3: $$\{c_1, ..., c_K\} \leftarrow \texttt{Calculer\_Clusters}(DE, VT)$$

4: **for** $q \in \{1, ..., K\}$ **do**

5: $\nu_q \leftarrow \texttt{Barycentre\_Riemannien}(c_q)$

6: **end for**

7: **for** $u \in DT$ **do**

8: $$Cluster(u) \leftarrow argmin_{p \in \{1,...,K\}}(d(\nu_p, u))$$

▷ d est la distance riemannienne

9: **end for**

10: **return** $Cluster$

associant par la suite une donnée de l'ensemble de test à une classe en utilisant la règle de Bayes (voir règle (69) dans [21]).

4 Expérimentations

Dans cette section, nous illustrons l'intérêt des approches proposées précédemment par quelques applications en apprentissage supervisé et non supervisé sur des graphes de référence. Nous montrons le gain apporté vis-à-vis de l'état de l'art en nous concentrant sur l'algorithme des K-moyennes. Les datatsets utilisés sont publics [20] et leurs caractéristiques sont données dans le tableau suivant (le nombre de nœuds, arrêtes et classes est désigné respectivement par Nœuds, Liens et K).

Datasets	Noeuds	Liens	K
Karate	34	77	2
Polblogs	1224	16781	2
Polbooks	105	441	3
Football	115	613	12
Adjnoun	112	425	2
Mammifères	1179	6541	NA

TABLE 1 – Caractéristiques des graphes utilisés.

4.1 Apprentissage non supervisé

Dans une première série d'expériences, nous avons utilisé les 5 premiers datasets donnés dans le tableau précédent. Pour chaque dataset, nous avons généré 10 marches aléatoires sur le graphe en suivant DeepWalk [19]. Sur chaque marche aléatoire, nous générons un plongement de Poincaré et un plongement euclidien dans $\mathbb{R}^{10}$ (en utilisant la version euclidienne de l'algorithme (3)). Ensuite, nous appliquons à chaque plongement hyperbolique (resp. euclidien), un algorithme des K-moyennes hyperbolique (resp. euclidien). Dans les deux cas, nous sélectionnons comme meilleur plongement celui ayant une variance totale minimale au sens de l'algorithme (3). Enfin, nous évaluons la performance des meilleurs plongements dans les deux cas ainsi que la performance moyenne sur les 10 tests effectués. Les résultats sont donnés dans le tableau 2 en utilisant les abréviations suivantes :

- MPP : meilleur plongement de Poincaré
- MPE : meilleur plongement euclidien
- 10MP : moyenne sur les 10 plongements de Poincaré
- 10ME : moyenne sur les 10 plongements euclidiens

Le tableau 2 montre que suivre un plongement par un algorithme des K-moyennes est plus avantageux dans le cas hyperbolique que dans le cas euclidien. La figure 1 montre l'évolution croissante de la performance MPP avec le nombre d'expérimentation effectués NE pour le dataset Football et illustre ainsi l'intérêt de générer plusieurs plongements.

En ce qui concerne l'exploitation des plongements de Poincaré pour le partitionnement des noeuds, remarquons que le meilleur plongement de Poincaré (MPP) tel que défini précédemment, n'obtient pas nécessairement la meilleure performance. Pour un seul dataset, la performance moyenne (10MP) obtenue est légèrement supérieure à MPP. Ceci peut être expliqué par l'existence de données aberrantes (des personnes pour lesquelles il est difficile de prédire une classe). L'avantage du meilleur plongement hyperbolique est remarquable pour l'exemple Football (+17%) et pour l'exemple Polbooks (+4.8%). En comparaison avec l'état de l'art

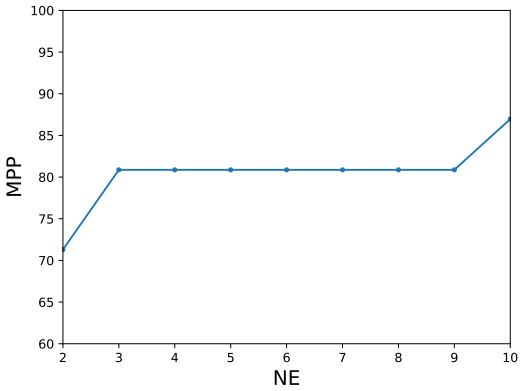


FIGURE 1 – Courbe représentant la variation du MPP en fonction de NE (nombre d'expérimentations effectuées)

récent, nos résultats dépassent ceux obtenus dans [1] par un plongement sur une surface généralisée. Les datasets considérés dans [1] sont **Polbooks, Football, Adjnoun** avec des taux de réussite respectifs de 75%, 77% et 51% respectivement. Les améliorations pour ces exemples en utilisant notre méthode sont significatives : $+\mathbf{9.8\%}, +\mathbf{10\%}$ et $+\mathbf{0.8}\%$.

La figure 2 montre des visualisations des clusters calculés en utilisant le meilleur plongement de Poincaré (MPP) pour chaque dataset. Chaque cluster est représenté par une couleur et le barycentre est symbolisé par un carré.

Comme deuxième application, nous avons considéré un exemple de données hiérarchiques (sous forme d'arbre) qui est le sous-arbre des mammifères extrait de Wordnet traité comme un graphe. La figure 3 montre les clusters obtenus quand $K = 6$. Les barycentres sont représentés par des carrés. Enfin, la figure 4 montre explicitement certaines étiquettes des noeuds choisies aléatoirement. Un focus est mis sur les noeuds proches des barycentres d'une part et sur les frontières entre les différents clusters d'autre part.

Remarquons que les clusters calculés permettent de discerner entre différents types de mammifères. Par exemple le cluster bleu contient en majorité la famille des canidés alors que le cluster orange contient des mammifères de taille plus importante (lion, tigre, éléphant,...).

4.2 Apprentissage supervisé

Dans cette partie, nous exploitons la notion du barycentre riemannien pour effectuer de l'apprentissage

	Apprentissage non-supervisé					Apprentissage supervisé	
Datasets	**MPP**	**MPE**	**10MP**	**10ME**	[1]	**10VC**	[11]
Karate	**91.2**%	70.6%	**91.4**%	65.8%	–	**93.9**%	86%
Polblogs	**92.8**%	51.9%	**92.5**%	53.5%	–	**92.4**	**93**%
Polbooks	**84.8**%	77.1%	**80**%	62%	75%	**83.3**%	73%
Football	**87%**	67.8%	**69.4**%	56.8%	77%	**77.9** %	24%
Adjnoun	51.8%	51.8 %	**52.5**%	51.6%	51%	**57.8** %	-

Table 2 – Tableau comparatif des performances obtenues pour les algorithmes d'apprentissage supervisé et non-supervisé. Les meilleurs résultats sont en gras.

supervisé sur les graphes en suivant l'algorithme 4. Pour évaluer la méthode, nous avons subdivisé chaque dataset de la liste précédente (wordnet exclu) en 5 parties : 4/5 pour l'entraînement et 1/5 pour le test. Après plongement de tout le graphe, chaque élément de l'ensemble test est associé à un cluster suivant la règle du plus proche barycentre. Nous avons répété cette expérience 5 fois en permutant les ensembles tests par validation croisée. La même expérience (5 permutations de l'ensemble test) a été refaite 10 fois. Finalement, nous avons évalué la performance moyenne en utilisant la vérité terrain des données tests. Le tableau 2 présente les résultats obtenus. On utilisera l'abréviation suivante :
10VC : Performance moyenne par validation croisée sur les 10 expériences effectuées.

Nous allons maintenant comparer nos résultats avec ceux de [11] qui utilise une généralisation de la méthode SVM au disque de Poincaré. [11] considère les datasets **Karate, Polbooks, Football, Polblogs** et obtient des taux de réussite respectifs de $86\%, 73\%, 24\%, 93\%$ sur 5 expérimentations par validation croisée sur 5 plongements différents selon [10]. Nous avons obtenu des améliorations significatives pour les datasets Karate, Polbooks et Football de $+\mathbf{7.9\%}$, $+\mathbf{10.3\%}$, de $+\mathbf{53.9}\%$ avec une légère baisse de performance de -0.6% pour le dataset Polblogs.

5 Conclusion

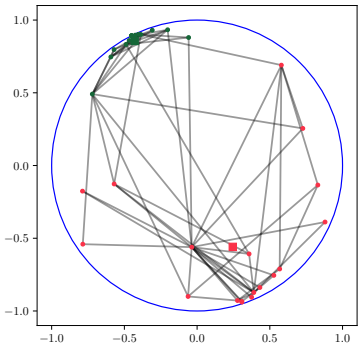
Dans cet article, nous avons présenté des algorithmes d'apprentissage sur les graphes basés sur les plongements de Poincaré et des extensions au cadre riemannien des algorithmes des K-moyennes et d'espérance-maximisation. Nous avons illustré l'intérêt des algorithmes des K-moyennes par des expérimentations en montrant le gain apporté vis-à-vis de l'état de l'art. D'autre part, nous avons présenté les algorithmes EM et esquissé quelques exemples de leurs applications en lien avec les plongements de Poincaré.

Notons qu'avec la notion de variance totale minimale introduite dans ce papier, il est toujours possible d'augmenter le nombre de plongements (fixé à 10 dans l'article) et aussi la dimension de l'espace de Poincaré (fixé à 2 dans l'article). Ce raisonnement est possible grâce aux bonnes propriétés des variétés hyperboliques et est complètement non supervisé dans le sens où il ne nécessite aucune vérité terrain. Par conséquent, des améliorations des résultats présentés dans ce papier sont toujours possibles.

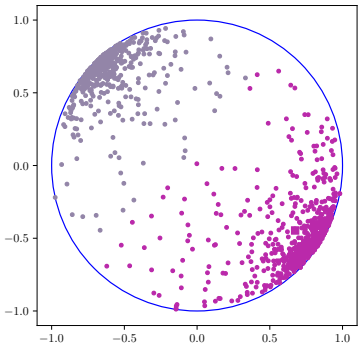
Les résultats obtenus avec le plongement euclidien utilisant DeepWalk suggèrent que pour obtenir de bonnes performances avec cette approche (ou autres variétés comme Graph2vec [16], Node2vec [12]), il faudrait prendre des espaces euclidiens avec des dimensions très grandes.

En perspective, nous envisageons d'étudier la complexité des algorithmes riemanniens présentés dans cet article. Nous comptons aussi utiliser ces travaux pour étendre les résultats de [9, 28] qui propose d'apprendre les communautés sur les graphes en utilisant les lois de mélange gaussien (euclidien). Notre proposition sera d'étendre cela à une version géométrique. Plus précisément au lieu de générer plusieurs plongements et en choisir un à la fin, il sera possible d'optimiser le plongement et d'entraîner le partitionnement par le mélange gaussien simultanément.

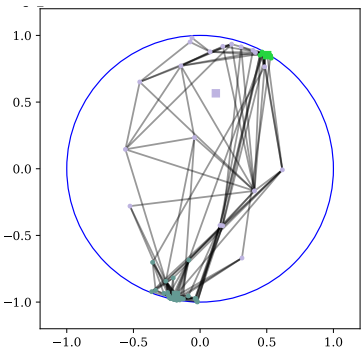
Enfin, nous souhaitons étendre les travaux à l'apprentissage en ligne et adresser le problème des tests statistiques basés sur les lois gaussiennes et leurs applications sur les données graphiques et les données texte.



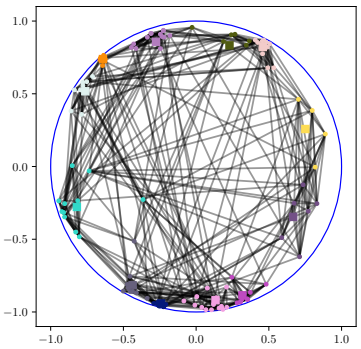
(a) Karate.



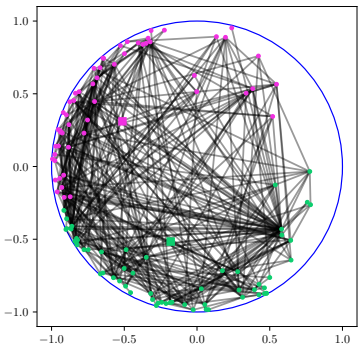
(b) Polblogs.



(c) Polbooks.



(d) Football.



(e) Adjnoun.

FIGURE 2 – Visualisation des clusters dans le meilleur plongement de Poincaré

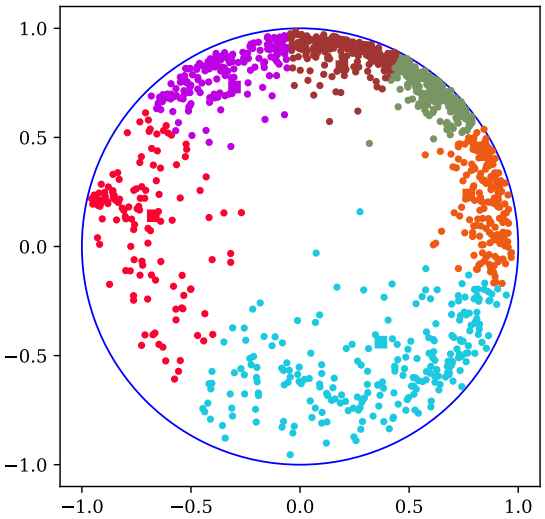
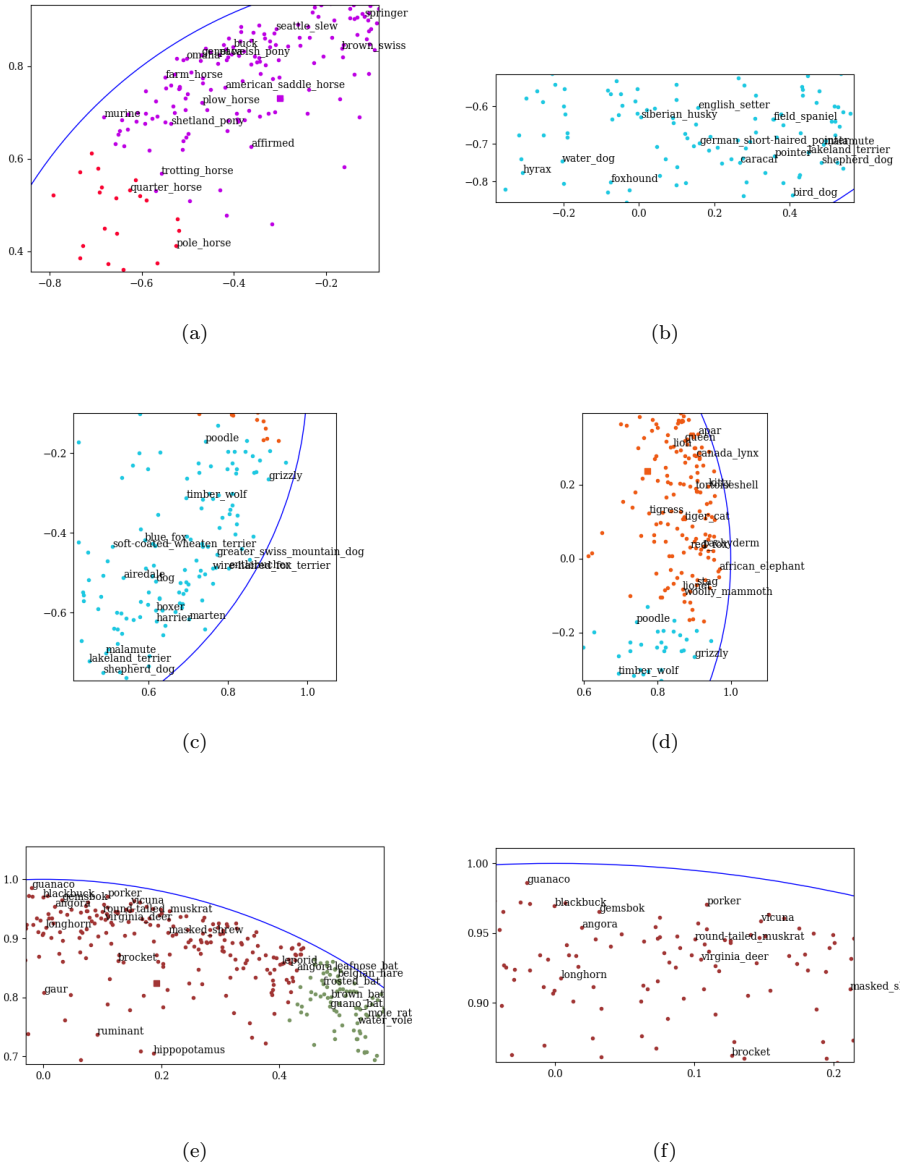


FIGURE 3 – Partitionnement du sous-arbre des mammifères en 6 clusters par l'algorithme des K-moyennes riemannien.



(a) (b) (c) (d) (e) (f)

FIGURE 4 – Aperçu des étiquettes du sous-arbre des mammifères : sur les frontières de clusters distincts et à l'intérieur des clusters dans un voisinage du barycentre (ce dernier est représenté par un carré)

Références

[1] M. Aalto and N. Verma. Metric learning on manifolds. *CoRR*, https ://arxiv.org/abs/1902.01738, 2018.

[2] B. Afsari. Riemannian L^p center of mass : existence, uniqueness and convexity. *Proc. Amer. Math. Soc.*, 139(2) :655–6673, 2011.

[3] M. Arnaudon, F. Barbaresco, and L. Yang. Riemannian medians and means with applications to radar signal processing. *J. Sel. Topics Signal Processing*, 7(4) :595–604, 2013.

[4] M. Arnaudon, C. Dombry, A. Phan, and L. Yang. Stochastic algorithms for computing means of probability measures. *Stoch. Proc. Appl.*, 58(9) :1473–1455, 2012.

[5] M. Arnaudon and L. Miclo. Means in complete manifolds : uniqueness and approximation. *ESAIM Probability and statistics*, 18 :185–206, 2014.

[6] M. Arnaudon, L. Yang, and F. Barbaresco. Stochastic algorithms for computing p-means of probability measures, geometry of Radar Toeplitz covariance matrices and applications to HR Doppler processing. In *International Radar Symposium (IRS)*, pages 651–656, 2011.

[7] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten. Multiclass brain-computer interface classification by Riemannian geometry. *IEEE Trans. Biomed. Engineering*, 59(4) :920–928, 2012.

[8] S. Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Trans. Autom. Control.*, 122(4) :2217–2229, 2013.

[9] S. Cavallari, V. W. Zheng, H. Cai, K. C. Chang, and E. Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 377–386, 2017.

[10] B. P. Chamberlain, J. Clough, and M. P. Deisenroth. Neural embeddings of graphs in hyperbolic space. *13th international workshop on mining and learning with graphs. Available at https ://arxiv.org/abs/1705.10359*, 2017.

[11] H. Cho, B. Demeo, J. Peng, and B. Berger. Large-margin classification in hyperbolic space. *CoRR*, abs/1806.00437, 2018.

[12] A. Grover and J. Leskovec. node2vec : Scalable feature learning for networks. *KDD : proceedings. International Conference on Knowledge Discovery & Data Mining*, 2016 :855–864, 2016.

[13] H. Hajri, H. Zaatiti, and G. Hebrail. Learning graph-structured data using Poincaré embeddings and Riemannian k-means algorithms. *submitted*, 2019.

[14] S. Helgason. *Differential geometry, Lie groups, and symmetric spaces.* American Mathematical Society, 2001.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[16] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. graph2vec : Learning distributed representations of graphs. *CoRR*, abs/1707.05005, 2017.

[17] M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems 30*, pages 6338–6347. Curran Associates, Inc., 2017.

[18] J. Pennington, R. Socher, and C. D. Manning. Glove : Global vectors for word representation. In *In EMNLP*, 2014.

[19] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk : Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.

[20] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[21] S. Said, L. Bombrun, Y. Berthoumieu, and J. H. Manton. Riemannian gaussian distributions on the space of symmetric positive definite matrices. *IEEE Trans. Information Theory*, 63(4) :2153–2170, 2017.

[22] S. Said, H. Hajri, L. Bombrun, and B. C. Vemuri. Gaussian distributions on Riemannian symmetric spaces : Statistical learning with structured covariance matrices. *IEEE Trans. Information Theory*, 64(2) :752–772, 2018.

[23] F. Sala, C. D. Sa, A. Gu, and C. Ré. Representation tradeoffs for hyperbolic embeddings. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 4457–4466, 2018.

[24] A. Tifrea, G. Bécigneul, and O.-E. Ganea. Poincaré glove : Hyperbolic word embeddings. *arXiv preprint arXiv :1810.06546*, 2018.

[25] C. Tu, H. Wang, X. Zeng, Z. Liu, and M. Sun. Community-enhanced network representation learning for network analysis. *CoRR*, abs/1611.06645, 2016.

[26] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1225–1234, New York, NY, USA, 2016. ACM.

[27] H. Zaatiti et al. Learning community embedding with Riemannian expectation maximisation algorithms. *Work in progress*, 2019.

[28] V. W. Zheng, S. Cavallari, H. Cai, K. C. Chang, and E. Cambria. From node embedding to community embedding. *CoRR*, abs/1610.09950, 2016.

Apprentissage multi-vues pour la complétion transmodale de matrices de noyaux
Cross-View Kernel Transfer

Riikka Huusari[1], Cécile Capponi[1], Hachem Kadri[1], et Paul Villoutreix[1,2]

[1]Aix-Marseille Univ., Université de Toulon, CNRS, LIS, Qarma
[2]Turing Center for Living Systems (CENTURI)

May 24, 2019

Résumé

Nous considérons le problème de complétion de matrices de noyaux dans un contexte d'apprentissage multivues. Une vue sous laquelle certaines données sont manquantes crée des colonnes et des lignes manquantes dans la matrice de noyau de cette vue. Pour résoudre ce problème, nous proposons un algorithme d'alignement de noyaux qui complète les informations manquantes dans chaque matrice noyau d'une vue en transférant la connaissance depuis les autres vues. Nous illustrons les avantages de notre approche avec des données simulées, ainsi qu'avec un jeu de données biologiques issues d'études sur la formation de motifs dans l'embryogénèse précoce de *Drosophila melanogaster*.

Mots-clef : Apprentissage multi-vues, complétion de matrices de noyaux, apprentissage de noyaux, données manquantes.

Abstract

We consider kernel completion problem with the presence of multiple views in the data. In this context the data samples can be fully missing in some views, creating missing columns and rows to the kernel matrices that are calculated individually for each view. We propose to solve this problem by transferring the features of the other views to represent the view under consideration. We align the known part of the kernel matrix with a new kernel built from the features of other views. We are thus able to find generalizable structures in the kernel under completion, and represent it accurately. The missing values can be predicted with the data available in other views. We illustrate the benefits of our approach with simulated data, as well as with real biological datasets from studies of pattern formation in early *Drosophila melanogaster* embryogenesis.

Keywords: Multi-view learning, cross-view transfer, kernel completion, kernel learning, missing data.

1 Introduction

Multi-view learning is a machine learning paradigm referring to a learning situation where data comtains various, often heterogenous, modalities that might be obtained from different sources or by different measurement techniques [SMDW19]. For example a data set might contain images with captions, both describing the same data sample but from different point of view. Learning by taking into account all the views and their interactions is expected to give better results than learning from each single view independently, because views are likely to carry complementary information and regularities.

Gathering multi-view data can be very expensive, and in some situations (such as some biological applications, or medical diagnosis from several physical examination devices) it might be outright impossible to simultaneously measure all the views under investigation. A typical example of the latter situation arises in developmental biology when several variables are of interest but cannot be measured simultaneously [VAL+17] or when results of heterogeneous types of experiments, such as spatial information and single cell transcriptomics, need to be integrated in a common representation [KWA+17]. Unfortunately many suc-

cessful multi-view methods cannot directly cope with data missing from the views. The simplest approach would be to neglect the samples with missing views, but depending on the amount of these samples this might make the data set so small as to make applying many of these machine learning methods non-feasible. Thus a preprocessing step to fill in the missing values is needed.

Kernel-based methods in multi-view context are widely used for example in computational biology as well as computer vision [L+09, PWCN02]. One succesful and widely applied set of methods is called Multiple Kernel Learning (MKL) [GA11]. In kernel methods, the data samples are not considered as is by the learning algorithm, but rather via a kernel function that takes two data samples and acts as a kind of similarity measure between them. This can be an especially advantageous property, as kernel functions can be defined for many types of data. For example, graphs can be difficult for some machine learning algorithms to handle, but kernel-based methods are able to treat them with no difficulty. Thus, in this framework it is natural to directly complete the kernels themselves instead of the original missing data. Kernel completion in multi-view setting is an emerging topic which has not been much investigated so far [Bha19].

Existing matrix completion methods can be applied to kernel completion on multi-view setting when only some individual kernel values are missing, and not the whole rows and columns. However when this is not the case and data samples contain complete missing views, these approaches cannot cope with completion task, and the multi-view structure of the data should be leveraged for kernel completion. First works for completing kernels of multiple views contain relatively restrictive assumptions, requiring one complete observed view [TAA03, TRDID10]. Going beyond this assumption, [RLK17] proposed an EM-algorithm that minimizes the KL-divergence of all the individual view matrices to their linear combination. Lastly, a framework for completing kernel matrices in multi-view setting has been proposed in [BKR17], where both within- and between-view relationships are considered in solving the problem. As within-view relationship they learn a low-rank approximation of the kernel based on the available values there, while the between-view relationship strategy is based on finding a set of related kernels for each missing entry and modeling the kernel as a weighted sum of those.

In this paper we propose a novel method for problem of multi-view kernel completion, that is based on idea of information transfer across the views. One assumption in multi-view learning is that there are some relationships between the views; the views are connected and they describe the same data, they are not fully independent. In our method we learn and transfer the information that other views contain to represent the view we wish to complete. We consider the features of the other views and align their transformation to known values we have in the kernel of the view we wish to complete, using the notion of kernel alignment [CSTEK02, CMR10]. When we have learned this transformation, we can predict the missing values based on the information in the other views. Our method is a very general in the sense that we do not require any of the views to be complete; all of them may have some missing data.

As a kernel learning method, ours resembles [PTKY11] studying domain adaptation problem, where a linear transformation similar to ours was applied to the kernel matrix, and learned. In their work they fixed the features to be transformed to be empirical features obtained from kernel matrix, and instead of optimizing with respect to kernel alignment they considererd Hilbert-Schmidt independence criterion [GBSS05]. In contrast to our wok, the idea of transforming the features was considered in the context of domain adaptation, where the goal was to learn a common feature representation given kernel containing data from two domains. In our case the transfer is done from multiple feature representations to one that describes still another kernel.

This paper is organized as follows. The next section introduces relevant background about kernels. Section 3 introduces our algorithm (called CVKT for Cross-View Kernel Transfer), which we validate with experiments on simulated and real data in section 4.3 before concluding.

2 Background

In this section we introduce relevant background of kernel methods, and the notation we use in this paper. We consider multi-view data $x \in \mathcal{X} = \mathcal{X}^{(1)} \times ... \times \mathcal{X}^{(V)}$ such that each (complete) data point is observed in V views, $x = (x^{(1)}, ..., x^{(V)})$.

In machine learning kernel methods are a very succesfull group of methods used in various tasks [HSS08]. The power using a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ in a learning task comes from the fact that it corresponds to an inner product in some feature space, that is,

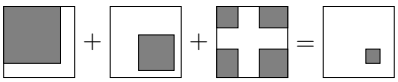
$$k(x, z) = \langle \phi(x), \phi(z) \rangle_{\mathcal{H}} .$$

This allows one to map data inexpensively to some (possibly infinite-dimensional) feature space where the data is expected to be better represented. In kernel-based learning algorithms the data is always delt with via the kernel function so this feature representation is never explicitely needed. In practice a matrix, $\mathbf{K}$, is built with the kernel function applied to each data sample such that $\mathbf{K}_{ij} = k(x_i, x_j)$.

For multi-view learning the simplest and most widely used kernel-based approach is to build the kernel as a combination of kernels from individual views. This combination is usually a weighted sum [GA11],

$$k(x, z) = \sum_{v=1}^{V} \alpha_v \, k^{(v)} \left(x^{(v)}, z^{(v)} \right), \tag{1}$$

where the weights α_v are often learned (MKL). Whenever there is some missing data in the views, obviously the sum cannot be calculated and the corresponding values in the final kernel matrix will be missing, too, as illustrated below, where grey areas of the kernel matrices indicate that the values are available, and white areas thus missing.



The goal of our work is to fill in these missing values in the kernel matrices by using the multi-view properties of the data, and leveraging the information contained in the other views in completing the missing values of a view.

Our algorithm uses the notion of kernel alignment [CSTEK02, CMR10] as the error measure, which is a similarity measure for comparing two kernel matrices. Alignment between two matrices $\mathbf{M}$ and $\mathbf{N}$ is defined as

$$A(\mathbf{M}, \mathbf{N}) = \frac{\langle \mathbf{M}_c, \mathbf{N}_c \rangle_F}{\|\mathbf{M}_c\|_F \|\mathbf{N}_c\|_F}, \tag{2}$$

where subscript c refers to centered matrices, that is, $\mathbf{M}_c = \mathbf{CMC}$ where $\mathbf{C} = \left[\mathbf{I}_n - \frac{1}{n}\mathbb{1}_n\mathbb{1}_n^\top\right]$ with $\mathbf{I}_n$ the identity matrix, and $\mathbb{1}_n$ vector of ones, and $\mathbf{M}$ is of size $n \times n$. In classification or regression setting the kernel to be learned is often aligned with an ideal kernel calculated with the known labels ($\mathbf{y}\mathbf{y}^\top$), as this is expected to produce good predictors [CMR10].

3 Algorithm

We propose to fill in the missing values in multi-view kernel matrices by transferring the information available in other views to represent the view in question. It is important to note that we do not assume that the views used in completing the other are fully observed; we only assume that each data sample is fully observed in at least one view whichever it is.

Given a multi-view data set $X^{(1)}, ..., X^{(V)}$ containing n samples, we can build $n \times n$ kernel matrices for each of the views, $\mathbf{K}^{(1)}, ..., \mathbf{K}^{(V)}$ that are then used in solving the learning problem. If we know the feature map explicitely, we can stack the elements $\phi(x_i)$ into a matrix, and get $\mathbf{K}^{(v)} = \Phi^{(v)}[\Phi^{(v)}]^\top$. For example with linear kernel we would have $\Phi^{(v)} = \mathbf{X}^{(v)}$ and $\mathbf{K}^{(v)} = \mathbf{X}^{(v)}[\mathbf{X}^{(v)}]^\top$. Of course this is not possible with some kernels, such as RBF kernel, whose feature map is infinite-dimensional. However the $\Phi^{(v)}$ is not unique, for example the empirical feature map [SMB+99] defined as $\mathbf{K}^{(v)}(\mathbf{K}^{(v)})^{-1/2}$ is equally valid choice and produces the same kernel matrix, since

$$\begin{aligned} &\mathbf{K}^{(v)}[\mathbf{K}^{(v)}]^{-1/2}[\mathbf{K}^{(v)}]^{-1/2}\mathbf{K}^{(v)} \\ &= \mathbf{K}^{(v)}[\mathbf{K}^{(v)}]^{-1}\mathbf{K}^{(v)} = \mathbf{K}^{(v)}. \end{aligned}$$

It is also possible to approximate this feature map, for example through Nyström approximation scheme [DM05] which is widely used in approximating kernel matrices. Nyström approximation is obtained by randomly sampling $p < n$ data samples, and with those calculating $\mathbf{K}^{(v)} \approx \mathbf{K}^{(v)}_{:,P}[\mathbf{K}^{(v)}_{P,P}]^{-1}\mathbf{K}^{(v)}_{P,:}$ where subscript P denotes the set of these p samples. In this case $\mathbf{K}^{(v)} \approx \Phi^{(v)}[\Phi^{(v)}]^\top$.

The kernel matrix $\mathbf{K}^{(v)}$ contains missing rows and columns if some of the data is missing for this view. We denote the set of indices where data is available for view v as $\mathcal{I}^{(v)}$, and the size of the set as $i^{(v)} \leq n$. Whenever clear from the context which view is in question we might leave the superscript out, denoting $\mathcal{I}^{(v)} = \mathcal{I}$. We denote the section of the kernel matrix of view v containing the known values as $\mathbf{K}^{(v)}_{\mathcal{I}}$; this is a matrix of size $i^{(v)} \times i^{(v)}$.

We propose to learn to represent kernel $\mathbf{K}^{(v)}_{\mathcal{I}}$ with the features of other views, and their interactions. This way we can somehow use the data available in other views for predicting the missing values of $\mathbf{K}^{(v)}$. To transfer the knowledge from other views towards the view v under question, we firstly build a large feature matrix from the feature matrices of the other views as follows

$$\Psi^{(v)}_{\mathcal{I}} = \left[\Phi^{(1)}_{\mathcal{I}^{(v)}}, ..., \Phi^{(v-1)}_{\mathcal{I}^{(v)}}, \Phi^{(v+1)}_{\mathcal{I}^{(v)}}, ..., \Phi^{(V)}_{\mathcal{I}^{(v)}}\right]. \tag{3}$$

From each view we take the samples that are available in view in question, $\mathcal{I}^{(v)}$. The new feature matrix $\Psi^{(v)}_{\mathcal{I}}$ is thus of size $i^{(v)} \times m^{(1)} + ... + m^{(v-1)} + m^{(v+1)} + ... + m^{(v)}$. This procedure is illustrated in Figure 1.

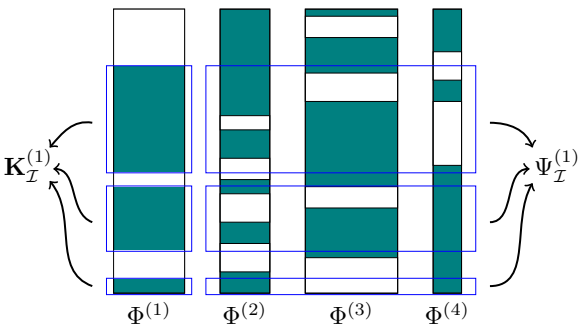


Figure 1: Illustration on building the feature matrix $\Psi_{\mathcal{I}}^{(v)}$ in our method from the feature representations $\Phi^{(v)}$. The white areas in $\Phi^{(v)}$ represent the missing data, and are filled with zero-inputation.

Learning to represent the target kernel $\mathbf{K}_{\mathcal{I}}^{(v)}$ with $\Psi_{\mathcal{I}}^{(v)}$ is done by considering a linear transformation of these features to some other feature space. This transformation is defined by matrix $\mathbf{U}^{(v)}$ of size $m^{(1)} + ... + m^{(v-1)} + m^{(v+1)} + ... + m^{(v)} \times r$. We wish to learn the optimal transformation $\mathbf{U}^{(v)}$ such that the transfer kernel $\Psi_{\mathcal{I}}^{(v)}\mathbf{U}^{(v)}[\Psi_{\mathcal{I}}^{(v)}\mathbf{U}^{(v)}]^\top$, is maximally aligned to the target kernel, giving us the optimization problem

$$\max_{\mathbf{U}^{(v)} \in S} \quad A\left(\mathbf{K}_{\mathcal{I}}^{(v)}, \Psi_{\mathcal{I}}^{(v)}\mathbf{U}^{(v)}\left[\Psi_{\mathcal{I}}^{(v)}\mathbf{U}^{(v)}\right]^\top\right), \qquad (4)$$

where we regularize the transformation matrix $\mathbf{U}^{(v)}$ by constraining it to the sphere manifold S, meaning that $\|\mathbf{U}^{(v)}\|_F = 1$. The optimization problem can be solved with gradient-based approach. We implemented this with the Pymanopt package [TKW16].[1]

After solving this optimization problem, a prediction on the full kernel matrix can be done via selecting all the other views to $\Psi^{(v)}$ as

$$\Psi^{(v)} = \left[\Phi^{(1)}, ..., \Phi^{(v-1)}, \Phi^{(v+1)}, ..., \Phi^{(V)}\right] \qquad (5)$$

and calculating

$$\tilde{\mathbf{K}}^{(v)} = \Psi^{(v)}\mathbf{U}^{(v)}\left[\Psi^{(v)}\mathbf{U}^{(v)}\right]^\top. \qquad (6)$$

We summarize the Cross-view Kernel Transfer (CVKT) procedure in Algorithm 1.

Compared to only two other approaches for multi-view kernel matrix completion [BKR17, RLK17], CVKT is much more computationally efficient in the sense that the optimization is not performed jointly over all the views. Instead we can treat the views and their respective completion problems independently, and gain in performance. The complexity of the algorithm is naturally dependent on the number of samples available in the view processed at each iteration, $i^{(v)}$, meaning that our algorithm is more efficient with more missing data. The other two important parameters, $d^{(v)}$ for the feature dimensions, and r for the number columns in $\mathbf{U}^{(v)}$ can be pre-set or cross-validated.

[1]The CVKT code will be made available upon acceptation of this paper, or by request before that.

Algorithm 1 CVKT algorithm

Require: Set of kernels $\mathbf{K}^{(1)}, ..., \mathbf{K}^{(V)}$;
set of indices of known values $\mathcal{I}^{(1)}, ..., \mathcal{I}^{(V)}$;
parameter r to control the size of feature transformation matrices $\mathbf{U}^{(v)}$

for $v \in [1, ..., V]$ **do**
  Calculate feature representation $\Phi_{\mathcal{I}}^{(v)}$ from $\mathbf{K}_{\mathcal{I}}^{(v)}$
end for
for $v \in [1, ..., V]$ **do**
  Build $\Psi_{\mathcal{I}}^{(v)}$ and $\Psi^{(v)}$ as in Eq. 3 and 5
  Solve for $\mathbf{U}^{(v)}$ in Eq. 4
  Predict $\tilde{\mathbf{K}}^{(v)}$ with $\Psi^{(v)}$ and $\mathbf{U}^{(v)}$ as in Eq. 6
end for
return $\tilde{\mathbf{K}}^{(1)}, ..., \tilde{\mathbf{K}}^{(V)}$

4 Experiments

In this section we empirically validate our approach (**CVKT**), first with simulated data, and then with a real dataset from study of pattern formation in *Drosophila melanogaster* embryogenesis. We compare our method to two simple baselines; **mean** and **zero imputation**, where the missing values are replaced with kernel mean value, or zeros, respectively.

We also consider the more elaborate **MKC** [BKR17] method, and use the code provided. [2] From the methods introduced in the paper, we focus on MKCemdbht, as the others were too slow to run.

4.1 Experimental protocols

In all CVKT experiments we use features extracted with Nyström approximation, and cross-validate over different approximation levels (20%, 40%, ..., 100%). We also cross-validate over the rank (or number of columns r) of matrices $\mathbf{U}^{(v)}$, over similar intervals (20%, 40%, ..., 100% of the full rank). For MKC, we

[2]`https://github.com/aalto-ics-kepaco/MKC_software`

performed the cross-validation over the same parameters as in [BKR17].

For measuring the kernel completion performance, we consider the metrics in the two other multi-view kernel matrix completion papers; the *completion accuracy (CA)* in Rivero et al [RLK17] and *average relative error (ARE)* in Bhadra et al [BKR17]. Additionally we consider the structural similarity index. The CA error measure is defined as

$$CA = \frac{1}{V}\sum_{v=1}^{V}\left(1 - \frac{\mathrm{Tr}\left(\mathbf{K}_{true}^{(v)}\mathbf{K}_{pred}^{(v)}\right)}{\left\|\mathbf{K}_{true}^{(v)}\right\|_F \left\|\mathbf{K}_{pred}^{(v)}\right\|_F}\right), \tag{7}$$

and the ARE over one view as

$$ARE = \frac{1}{n^{(v)} - i^{(v)}} \sum_{t \notin \mathcal{I}^{(v)}} \frac{\left\|\mathbf{K}_{pred}^{(v)}[t,:] - \mathbf{K}_{true}^{(v)}[t,:]\right\|_2}{\left\|\mathbf{K}_{true}^{(v)}[t,:]\right\|_2}, \tag{8}$$

where $[t,:]$ refers to the row t of the kernel matrix. Unlike CA, the error measure ARE is only computed over the rows corrensponding to the originally missing samples. In both of these error measures lower value means better completion performance, for structural similarity the higher the better.

Our method is expected to find generalizable structures on the kernel and predicting them in the completed matrices. It is important to notice that while this is the case, the original known values of the kernel are not necessarily fully preserved in the learned kernel. Thus in experiments we perform post-processing on the kernel predicted with CVKT by scaling the kernel values to the range of values in original kernel matrix, and shifting it so that the mean is the same as in the known part of the original kernel.

4.2 Simulated data

To validate our algorithm and to illustrate its generalization properties in predicting kernel values, we performed experiments with a simulated data set. We have created 100 data samples with a simple vector autoregression model of memory 1 where we periodically change the parameters of the model evolution, and constructed 7 views from overlapping column groups of the matrix to which the time series vectors have been stacked into. We calculated RBF kernels from these views. We consider a missing data scenario where every data sample is missing from randomly selected a views, a ranging from 1 to 4.

We report the results averaged over all the views for the various levels of missing data in Table 1, where we compare it to the other methods. To highlight the difference of our method to mean imputation that also performs relatively well with respect to the error measures, we show an example of a completed kernel matrix in Figure 2. Our method learns the overall trends in the kernel matrices, and is able to predict and generalize those.

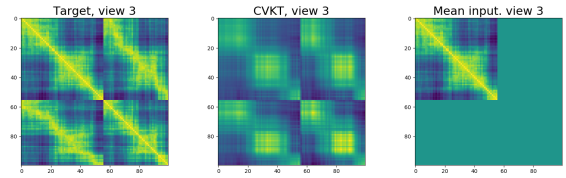


Figure 2: Target kernel matrix (left), our predicted kernel matrix (middle), and mean imputed kernel matrix (right) of view 3 in the scenario when 3 views samples are missing per data sample. The kernel matrices are reordered for better visualization such that top left corner contains the originally known data samples.

4.3 *Drosophila melanogaster* pattern formation data set

One highly relevant application of the cross-view kernel transfer method is in the field of developmental biology. Developmental biology is concerned with the study of how an embryo develops from a single fertilized cell into a complex and organized multicellular system [BWW+18]. This process involves dynamics at multiple scales which are recorded using numerous acquisition techniques, from live movies using fluorescent reporter proteins to fixed samples in *in situ* hybridization and immunocytochemistry techniques [MCP+16]. Since experimenting on human embryos can be difficult for ethical and practical reasons, several model organisms have been established over the years to explore the mechanisms of development. As an example, the *Drosophila melanogaster* embryo is one of the leading model organisms, because of its rapid development and its amenability to genetics experiments.

To study how cell fates are established by gene regulatory networks, it has recently been proposed that a first necessary step is to integrate multiple views from heterogeneous image datasets [VAL+17]. Gene regulatory networks describe the sequence of interaction between various chemical species inside a cell or within a tissue, which ultimately lead to cell differentiation into a variety of functional types. The number of vari-

Table 1: The kernel completion results on simulated data averaged over the seven views in the data with various amounts of missing views per data sample (a). Our result was postprocessed with a simple linear rescaling.

Error measure	a	CVKT	MKC	zero-input.	mean-input.
CA	1	**0.010±0.003**	0.071±0.065	0.143±0.039	0.015±0.006
	2	**0.012±0.002**	0.054±0.024	0.285±0.048	0.027±0.007
	3	**0.015±0.004**	0.114±0.058	0.427±0.049	0.038±0.009
	4	**0.025±0.002**	0.309±0.062	0.571±0.043	0.047±0.011
ARE	1	0.134±0.022	0.259±0.137	0.373±0.054	**0.123±0.028**
	2	**0.145±0.012**	0.254±0.050	0.530±0.044	0.178±0.029
	3	**0.156±0.027**	0.349±0.082	0.649±0.036	0.220±0.034
	4	**0.178±0.008**	0.567±0.052	0.752±0.026	0.256±0.038
S.sim	1	**0.701±0.035**	0.417±0.216	0.269±0.105	0.633±0.110
	2	**0.606±0.036**	0.326±0.097	0.106±0.032	0.480±0.072
	3	**0.516±0.026**	0.205±0.074	0.055±0.017	0.418±0.043
	4	0.385±0.048	0.072±0.025	0.030±0.006	**0.401±0.021**

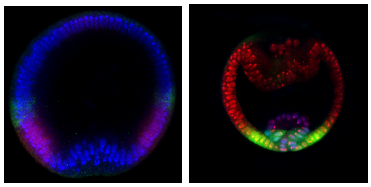


Figure 3: Examples of the images from the embryo dataset. The images are from datasets 1 (left) and 2 (right), where the colours show the views.

Table 2: Data availability in the views of *Drosophila melanogaster* data, 'x' referring to available data, '-' to missing; D to dataset and V to view.

	M	V1	V2	V3	V4	V5
D1	x	x	x	-	-	-
D2	x	x	-	x	x	-
D3	x	x	-	-	-	x
D4	x	-	x	x	-	x

ables in these networks can go up to hundreds and each of them have to be measured separately with specific reporters. To understand the kinetics of these interactions it is necessary to reconstruct the time courses of their levels in various parts of the embryo. Despite many advances in microscopy techniques, it still challenging to measure more than three of these variables at the same time, in addition, in the absence of reliable live reporters, some variables can only be measured in fixed images where the development is arrested, hence the need to integrate multiple views. As an illustration, live imaging of gastrulation provides information about nuclear positions as a function of time, but is silent about the levels of gene expression. On the other hand, an image of a fixed embryo reveals the distribution of an active enzyme but has no direct temporal information.

In the following example, we follow [VAL+17] and focus on the dorso-ventral patterning in *Drosophila melanogaster* early development. In this model system a graded profile of nuclear localization of a transcription factor named Dorsal (Dl) establishes the dorsoventral (DV) stripes of gene expression. Four datasets of fixed images were acquired to visualize nuclei (referred to as M, for morphology), protein expression of doubly phosophorylated ERK (dpERK, V1), Twist (V2), and Dorsal (V4), and mRNA expression of *ind* (V3) and *rho* (V5). The first dataset contains 108 images stained for dpERK and Twist. The second dataset contains 59 images stained for dpERK, *ind*, and Dorsal. The third dataset contains 58 images stained for dpERK, *ind*, and *rho*. The fourth dataset contains 30 images stained for Twist, *ind*, and *rho*. Examples of the images the data contains can be seen in Figure 3. The distribution of the variables are shown on Table 2.[3]

In order to quantify the success of the proposed

[3]the view *ind* of the third dataset remains unused because of the lesser quality of the staining [VAL+17].

Table 3: Kernel matrix completion results on embryo data set where 30% of available data is selected to be missing randomly per view.

Error measure	view	CVKT	MKC	zero-input.	mean-input.
CA	1	0.389	0.275	0.258	**0.208**
	2	0.297	0.311	0.236	**0.173**
	3	**0.112**	0.224	0.245	0.165
	4	**0.104**	0.156	0.231	0.146
	5	**0.141**	0.162	0.234	0.164
ARE	1	0.789	0.519	0.441	**0.399**
	2	0.693	0.648	0.414	**0.360**
	3	0.441	0.452	0.418	**0.345**
	4	0.419	0.338	0.395	**0.326**
	5	0.494	**0.349**	0.414	**0.350**
S.sim	1	0.475	0.676	**0.725**	0.617
	2	0.449	0.490	0.614	**0.691**
	3	**0.783**	0.539	0.633	0.655
	4	**0.739**	0.633	0.487	0.625
	5	0.625	**0.654**	0.625	0.588

method, we select randomly samples to be missing for each of the views. We then try to complete these samples with the information available in the other views. Note that we do not try in this setting to complete the really missing samples. Thus for example when we consider view 2, we will only deal with datasets 1 and 4 (see Table 2).

Our CVKT performs better in most of the views than other state-of-the-art methods with respect to CA error measure, shown in Table 3. Moreover, from Figure 4 we can see that the structure of the kernel matrix is learned very well; however the exact values in our learned kernel matrices are slightly different ("lighter" images), which is no doubt then seen in the error measures.

5 Conclusion

We have introduced a novel idea for performing multi-view kernel matrix completion by transferring cross-view knowledge to represent the views with missing values. We learn to represent the kernels with features of other views linearly transformed to a new feature space. This allows predicting the missing values of a kernel with features available in the other views. Our algorithm solves the problem efficiently, since the views can be treated individually, and no heavy joint optimization is performed. This individual treatement of views also gives more flexibility to our approach. As our experiments with simulated and real data demonstrate, our method is able to find generalizable structures from the incomplete kernel matrices, and is able to predict those structures in completing them. As a succesfull multi-view kernel completion method, this work opens up novel avenues of research also for the reconstruction of the initial data samples.

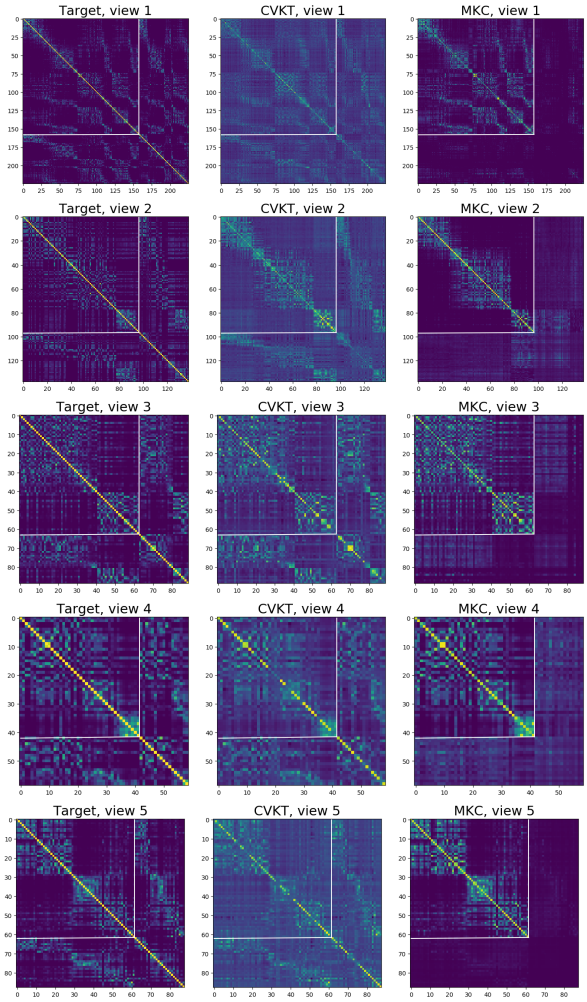


Figure 4: Target kernel matrices (left), our predicted kernel matrices (middle) with CVKT, and MKC predicted kernel matrices (right) of embryo data when randomly selected 30% of the available samples were set to be missing. The kernel matrices are reordered for better visualization such that top left corner contains the originally known data samples (areas with unknown and known samples are separated with white lines).

Acknowledgements

This work is granted by the french national project ANR Lives ANR-15-CE23-0026, and by the Turing Center for Living Systems (CENTURI) for PV.

References

[Bha19] Sahely Bhadra. Multi-view data completion. In *Linking and Mining Heterogeneous and Multi-view Data*, pages 1–25. Springer, 2019.

[BKR17] Sahely Bhadra, Samuel Kaski, and Juho Rousu. Multi-view kernel completion. *Machine Learning*, 106(5):713–739, 2017.

[BWW+18] James A Briggs, Caleb Weinreb, Daniel E Wagner, Sean Megason, Leonid Peshkin, Marc W Kirschner, and Allon M Klein. The dynamics of gene expression in vertebrate embryogenesis at single-cell resolution. *Science*, 360(6392):eaar5780, 2018.

[CMR10] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Two-stage learning kernel algorithms. In *ICML*, pages 239–246, 2010.

[CSTEK02] Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz S Kandola. On kernel-target alignment. In *NIPS*, pages 367–373, 2002.

[DM05] Petros Drineas and Michael W Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(Dec):2153–2175, 2005.

[GA11] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of machine learning research*, 12(Jul):2211–2268, 2011.

[GBSS05] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

[HSS08] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[KWA+17] Nikos Karaiskos, Philipp Wahle, Jonathan Alles, Anastasiya Boltengagen, Salah Ayoub, Claudia Kipar, Christine Kocks, Nikolaus Rajewsky, and Robert P Zinzen. The drosophila embryo at single-cell transcriptome resolution. *Science*, 358(6360):194–199, 2017.

[L+09] Christoph H Lampert et al. Kernel methods in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 4(3):193–285, 2009.

[MCP+16] Erik Meijering, Anne E Carpenter, Hanchuan Peng, Fred A Hamprecht, and Jean-Christophe Olivo-Marin. Imagining the future of bioimage analysis. *Nature biotechnology*, 34(12):1250, 2016.

[PTKY11] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[PWCN02] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Stafford Noble. Learning gene functional classifications from multiple data types. *Journal of computational biology*, 9(2):401–411, 2002.

[RLK17] Rachelle Rivero, Richard Lemence, and Tsuyoshi Kato. Mutual kernel matrix completion. *IEICE transactions on Information and Systems*, 100(8):1844–1851, 2017.

[SMB+99] Bernhard Schölkopf, Sebastian Mika, Chris JC Burges, Philipp Knirsch, Klaus-Robert Müller, Gunnar Rätsch, and Alexander J Smola. Input space versus feature space in kernel-based methods. *IEEE transactions on neural networks*, 10(5):1000–1017, 1999.

[SMDW19] Shiliang Sun, Liang Mao, Ziang Dong, and Lidan Wu. *Multiview Machine Learning*. Springer, 2019.

[TAA03] Koji Tsuda, Shotaro Akaho, and Kiyoshi Asai. The EM algorithm for kernel matrix completion with auxiliary data. *Journal of*

machine learning research, 4(May):67–81, 2003.

[TKW16] James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016.

[TRDID10] Anusua Trivedi, Piyush Rai, Hal Daumé III, and Scott L DuVall. Multi-view clustering with incomplete views. In *NIPS Machine Learning for Social Computing Workshop*, 2010.

[VAL+17] Paul Villoutreix, Joakim Andén, Bomyi Lim, Hang Lu, Ioannis G. Kevrekidis, Amit Singer, and Stanislav Y. Shvartsman. Synthesizing developmental trajectories. *PLOS Computational Biology*, 13(9):1–15, 09 2017.

Sélection d'attributs agnostique

Guillaume DOQUET and Michèle SEBAG

TAO, CNRS − INRIA − LRI, Université Paris-Saclay

Abstract

Unsupervised feature selection is mostly assessed along a supervised learning setting, depending on whether the selected features efficiently permit to predict the (unknown) target variable. Another setting is proposed in this paper : the selected features aim to efficiently recover the whole dataset. The proposed algorithm, called AgnoS, combines an auto-encoder with structural regularizations to sidestep the combinatorial optimization problem at the core of feature selection. The extensive experimental validation of AgnoS on the scikit-feature benchmark suite demonstrates its ability compared to the state of the art, both in terms of supervised learning and data compression.

1 Introduction

With the advent of big data, high-dimensional datasets are increasingly common, with possibly negative consequences on the deployment of machine learning algorithms in terms of i) computational cost; ii) accuracy (due to overfitting or lack of robustness related to e.g. adversarial examples (Goodfellow et al., 2015)); and iii) poor interpretability of the learned models.

The first two issues can be handled through dimensionality reduction, based on feature selection (Chandrashekar and Sahin, 2014; Nie et al., 2016; Chen et al., 2017; Li et al., 2018) or feature construction (Tenenbaum et al., 2000; Saul and Roweis, 2003; Wiatowski and Bölcskei, 2018). The interpretability of the learned models, an increasingly required property for ensuring *Fair, Accountable* and *Transparent* AI (Doshi-Velez and Kim, 2017), however is hardly compatible with feature construction, and feature selection (FS) thus becomes a key ingredient of the machine learning pipelines.

This paper focuses on *unsupervised feature selection.* Most FS approaches tackle supervised FS (Song et al., 2012; Chandrashekar and Sahin, 2014; Chen et al., 2017), aimed to select features supporting a (nearly optimal) classifier. Quite the contrary, unsupervised FS is not endowed with a natural learning criterion. Basically, unsupervised FS approaches tend to define pseudo-labels, e.g. based on clusters, and falling back on supervised FS strategies, aim to select features conducive to identify the pseudo labels (more in section 3). Eventually, unsupervised FS approaches are assessed within a supervised learning setting.

Following Y. LeCun's claim that unsupervised learning constitutes the bulk of machine learning, and that any feature might define an (auxiliary) learning goal, this paper tackles *Agnostic Feature Selection* with the goal of *leaving no feature behind.* Specifically, an unsupervised FS criterion aimed to select a subset of features supporting the prediction of *every* initial feature, is proposed. The proposed AgnoS approach combines Auto-encoders with structural regularization, and delegates the combinatorial optimization problem at the core of feature selection to a regularized data compression scheme (section 2).

The contribution of the paper is threefold. Firstly, three regularization schemes are proposed and compared to handle the redundancy of the initial data representation. Informally, if the feature set includes duplicated features, the probability of selecting *one* copy of this feature should increase; but the probability of selecting several copies of any feature should be very low at all times. Several types of regularizations are proposed and compared to efficiently handle feature redundancy: regularization based on slack variables (AgnoS-S); L_2-L_1 regularization based on the auto-encoder weights (AgnoS-W); and L_2-L_1 regularization based on the auto-encoder gradients (AgnoS-G).
A second contribution is to show on the scikit-feature benchmark (Li et al., 2018) that AgnoS favorably compares with the state of the art (He et al., 2005; Zhao and Liu, 2007; Cai et al., 2010; Li et al., 2012) considering the standard assessment procedure.

A third contribution is to experimentally show the brittleness of this standard assessment procedure, demonstrating that it does not allow to reliably compare unsupervised FS approaches (section 5).

The paper concludes with a discussion and some perspectives for further research.

Notations. In the following, the dataset is denoted $X \in \mathbb{R}^{n \times D}$, with n the number of samples and D the number of features. x_i (respectively f_j) denotes the i-th sample (resp. the j-th feature). The feature set is noted $F = (f_1, ..., f_D)$. $f_i(x_k)$ denotes the value taken by the i-th feature on the k-th sample.

2 AgnoS

The proposed approach relies on feature construction, specifically on Auto-Encoders, to find a compressed description of the data. As said, feature construction does not comply with the requirement of interpretability. Therefore, AgnoS will use an enhanced learning criterion to retrieve the initial features most essential to approximate *all features*, in line with the goal of *leaving no feature behind*.

This section is organized as follows. For the sake of self-containedness, the basics of Auto-Encoders are summarized in section 2.1. A key AutoEncoder hyperparameter is the dimension of the latent representation (number of neurons in the hidden layer), which should be set according to the intrinsic dimension (ID) of the data for the sake of information preserving. Section 2.2 thus briefly introduces the state of the art in ID estimation.

In order to delegate the feature selection task to the auto-encoder, the learning criterion is regularized to be robust w.r.t. redundant feature sets. A first option considers weight-based regularization along the lines of LASSO (Tibshirani, 1996) and Group-LASSO (Yuan and Lin, 2007) (section 2.4.1). A second option uses a regularization defined on the gradients of the encoder ϕ (section 2.4.2). A third option uses slack variables (section 2.5).

2.1 Auto-Encoders

Auto-Encoders (AE) are a class of neural networks designed to perform data compression via feature construction. The encoder ϕ and the decoder ψ are trained to approximate identity, i.e. such that for each training point x :

$$\psi \circ \phi(x) \approx x$$

in the sense of the Euclidean distance, where the dimension d of the hidden layer is chosen to avoid the trivial solution of $\phi = \psi = Id$. Formally,

$$\phi, \psi = \arg\min \sum_{i=1}^{n} \|x_i - \psi \circ \phi(x_i)\|_2^2$$

Letting f_i denote the i-th initial feature and $\hat{f}_i$ its reconstructed version, the mean square error (MSE) loss above can be rewritten as :

$$L(F) = \sum_{i=1}^{D} ||\hat{f}_i - f_i||_2^2 \quad (1)$$

The use of AE to support feature selection raises two difficulties. The first one concerns the setting of the dimension d of the hidden layer (more below). The second one is the fact that the MSE loss (Eq. 1) is vulnerable to the redundancy of the initial description of the domain: typically when considering duplicated features, the effort devoted by the AE to the reconstruction of this feature increases with its number of duplicates. In other words, the dimensionality reduction criterion is biased to favor redundant features.

2.2 Intrinsic dimension

A necessary condition for the auto-encoder to preserve the information in the data, is the dimension of the hidden layer in an AE to be at least as large as the *intrinsic dimension* (ID) of the data (necessary though not sufficient condition).

Estimating the ID of a dataset has been thoroughly studied in the ML and statistical physics literature (Levina and Bickel, 2005; Camastra and Staiano, 2016; Facco et al., 2017), notably in relation with data visualization (Maaten and Hinton, 2008; McInnes et al., 2018).

The best known linear ID estimation relies on Principal Component Analysis, usually considering the eigenvalues λ_i (with $\lambda_i > \lambda_{i+1}$) and computing d such that the top-d eigenvalues retain a sufficient fraction τ of the data inertia ($\sum_{i=1}^{d} \lambda_i^2 = \tau \sum_{i=1}^{D} \lambda_i^2$). Another approach is based on the minimization of the stress (Cox and Cox, 2000), that is, the bias between the distance of any two points in the initial representation, and their distance along a linear projection in dimension d. Nonlinear approaches such as Isomap (Tenenbaum et al., 2000) or Locally Linear Embedding (Saul and Roweis, 2003), respectively aim at finding a mapping on $\mathbb{R}^d$ such that it preserves the geodesic distance among points or the local barycentric description of the data.

The approach used in the following relies instead on the Poisson model of the number of points in the hypersphere in dimension d $\mathcal{B}(0,r)$, increasing like r^d (Levina and Bickel, 2005; Facco et al., 2017). Considering for each point x its nearest neighbor x' and its 2nd nearest neighbor $x"$, defining the ratio $\mu(x) = \|x-x'\|/\|x-x"\|$ and averaging μ over all points in the dataset, it comes (Facco et al., 2017):

$$d = \frac{\log(1 - H(\mu))}{\log(\mu)} \tag{2}$$

with $log(1 - H(\mu))$ the linear function associating to $log(\mu_i)$ its normalized rank among the $\mu_1, \dots \mu_n$ in ascending order.[1]

2.3 AgnoS

AgnoS proceeds like a standard AutoEncoder, with every feature being preliminarily normalized and centered. The encoder weight vector W is also normalized ($\|W\|_2 = 1$). As the dimension of the latent representation is no less than the intrinsic dimension of the data by construction, and further assuming that the neural architecture of the AutoEncoder is complex enough, the AE defines a latent representation capturing the information of the data to the best possible extent (Eq. 1).

The key issue in AgnoS is twofold. The first question is to extract the initial features best explaining the latent features; if the latent features capture all the data information, the initial features best explaining the latent features will be sufficient to recover *all* features. The second question is to address the feature redundancy and prevent the AE to be biased in favor of the most redundant features.

Two approaches have been considered to address both goals. The former one is inspired from the well-known LASSO (Tibshirani, 1996) and Group-LASSO (Yuan and Lin, 2007). These approaches are extended to the case of neural nets (below). The latter approach is based on a particular neural architecture, involving slack variables (section 2.5).

2.4 AgnoS with LASSO regularization

After first summarizing the basics of LASSO and group-LASSO, this section describes how their extension within the AutoEncoder framework supports feature selection.

2.4.1 LASSO and Group-LASSO

Considering a standard linear regression setting from the dataset $\{(x_i, y_i), x_i \in \mathbb{R}^D, y_i \in \mathbb{R}, i = 1 \dots n\}$, the goal of finding the best weight vector $\beta \in \mathbb{R}^D$ minimizing $\|y - \langle x, \beta\rangle\|^2$ is prone to overfitting in the large D, small n regime. To combat the overfitting issue, Tibshirani (1996) introduced the *LASSO* technique, which adds a L_1 penalization term to the optimization problem, parameter $\lambda \geq 0$ governing the severity of the penalization:

$$\beta^* = \arg\min_{\beta} ||\mathbf{y} - X\beta||_2^2 + \lambda||\beta||_1 \tag{3}$$

Compared to the mainstream L_2 penalization − which also combats overfitting −, the L_1 penalization acts as a sparsity constraint: every i-th feature with corresponding weight $\beta_i = 0$ can be omitted, and the L_1 penalization effectively draws the weight of many features to 0. Note that in counterpart the solution is no longer rotationally invariant (Ng, 2004).

The *group LASSO* (Yuan and Lin, 2007) and its many variants (Meier et al., 2008; Simon et al., 2013; Ivanoff et al., 2016) have been proposed to retain the sparsity property while preserving the desired invariances among the features. Let us consider a partition of the features in groups $G_1, \dots, G_k$ of equal size, the L_2-L_1 penalized regression setting reads:

$$\beta^* = \arg\min_{\beta} ||y - X\beta||_2^2 + \lambda \sum_{i=1}^{k} \sqrt{\sum_{j \in G_i} \beta_i^2} \tag{4}$$

where the L_1 part enforces the sparsity at the group level (as many groups are inactive as possible) while preserving the rotational invariance within each group.

2.4.2 AgnoS-W: with L_2-L_1 weight regularization

Under the assumption that all latent variables are needed to reconstruct the initial features (section 2.2), denoting $\phi(F) = (\phi_1 \dots, \phi_d)$ the encoder function, with $\phi_k = \sigma(\sum_{\ell=1}^{D} W_{\ell,k} f_\ell + W_{0,k})$ and $W_{i,j}$ the encoder weights, the impact of the i-th feature on the latent variables is visible through the weight vector $W_{i,\cdot}$. It thus naturally comes to define the L_2-L_1 penalization within the encoder as:

$$L(W) = \sum_{i=1}^{D} \sqrt{\sum_{k=1}^{d} W_{i,k}^2} = \sum_{i=1}^{D} \|W_{i,\cdot}\|_2$$

[1] That is, assuming with no loss of generality that

$$\mu_i < \mu_{i+1}$$

one approximates the curve $(log(1 - i/n), log(\mu_i))$ with a linear function, the slope of which is taken as approximation of d.

and the learning criterion of AgnoS-W (Alg. 1) is accordingly defined as:

$$L(F) = \sum_{i=1}^{D} ||\hat{f}_i - f_i||_2^2 + \lambda L(W) \quad (5)$$

with λ the penalization weight. The sparsity pressure exerted through penalty $L(W)$ will result in setting $W_{i,\cdot}$ to 0 whenever the contribution of the i-th initial variable is not necessary to reconstruct the initial variables, that is, when the i-th initial variable can be reconstructed from the other variables.

This learning criterion thus expectedly supports the selection of the most important initial variables. Formally, the score of the i-th feature is the maximum absolute value of $W_{i,j}$ for j varying in $1, \ldots D$:

$$Score_W(f_i) = \|W_{i,\cdot}\|_\infty \quad (6)$$

The rationale for considering the infinity norm of $W_{i,\cdot}$ (as opposed to its L_1 or L_2 norm) is based on the local informativeness of the feature (see also *MCFS* (Cai et al., 2010), section 3): the i-th feature matters as long as it has a strong impact on at least one of the latent variables.

The above argument relies on the assumption that all latent variables are needed, which holds by construction.[2]

Algorithm 1 AgnoS-W

Input : Feature set $F = \{f_1, ..., f_D\}$
Parameter: λ
Output : Ranking of features in F
Normalize each feature to zero mean and unit variance.
Estimate intrinsic dimension $\widehat{ID}$ of F.
Initialize neural network with $d = \widehat{ID}$ neurons in the hidden layer.
Repeat
Backpropagate $L(F) = \sum_{i=1}^{D} \|\hat{f}_i - f_i\|_2^2 + \lambda \sum_{i=1}^{D} \|W_{i,\cdot}\|_2$
until convergence
Rank features by decreasing scores with $Score_W(f_i) = \|W_{i,\cdot}\|_\infty$.

2.4.3 AgnoS-G: with L_2-L_1 gradient regularization

In order to take into account the overall flow of information from the initial variables f_i through the auto-encoder, another option is to consider the gradient of the encoder $\phi = (\phi_1 \ldots \phi_d)$. Varga et al. (2017); Alemu et al. (2018); Sadeghyan (2018) have recently highlighted the benefits of hidden layer gradient regularization for improving the robustness of the latent representation.

Along this line, another L_2-L_1 regularization term is considered:

$$L(\phi) = \sum_{i=1}^{D} \sqrt{\sum_{k=1}^{n} \sum_{j=1}^{d} \big(\frac{\partial \phi_j}{\partial f_i}(x_k)\big)^2}$$

and the learning criterion is likewise defined as:

$$L(F) = \sum_{i=1}^{D} ||\hat{f}_i - f_i||_2^2 + \lambda L(\phi) \quad (7)$$

The sparsity constraint now pushes toward cancelling all gradients of the ϕ_j w.r.t. an initial variable f_i. The sensitivity score derived from the trained auto-encoder, defined as:

$$\text{Score}_G(f_i) = \max_{1 \le j \le d} \sum_{k=1}^{n} \big(\frac{\partial \phi_j}{\partial f_i}(x_k)\big)^2 \quad (8)$$

is used to rank the features by decreasing score. The rationale for using the max instead of the average is same as for Score_W. Note that in the case of an encoder with a single hidden layer with *tanh* activation, one has:

$$\text{Score}_G(f_i) = \max_{1 \le j \le d} \sum_{k=1}^{n} \big(W_{i,j}(1 - \phi_j(x_k)^2)\big)^2 \quad (9)$$

Algorithm 2 AgnoS-G

Input : Feature set $F = \{f_1, ..., f_D\}$
Parameter: λ
Output : Ranking of features in F
Normalize each feature to zero mean and unit variance.
Estimate intrinsic dimension $\widehat{ID}$ of F.
Initialize neural network with $d = \widehat{ID}$ neurons in the hidden layer.
Repeat
Backpropagate $L(F) = \sum_{i=1}^{D} ||\hat{f}_i - f_i||_2^2 + \lambda \sum_{i=1}^{D} \sqrt{\sum_{k=1}^{n} \sum_{j=1}^{d} (\frac{\partial \phi_j}{\partial f_i}(x_k))^2}$
until convergence
Rank features by decreasing scores with $Score_G(f_i) = \max_{j \in [1,\ldots,d]} \sum_{k=1}^{n} \big(\frac{\partial \phi_j}{\partial f_i}(x_k)\big)^2$.

[2] A question however is whether all latent variables are equally important. It might be that some latent variables are more important than others, and if an initial variable f_i matters a lot for an unimportant latent variable, the f_i relevance might be low. Addressing this concern is left for further work.

2.5 AgnoS-S: with slack variables

A third version of AgnoS is considered, called AgnoS-S and inspired from Leray and Gallinari (1999); Kalainathan et al. (2018). The idea is to augment the neural architecture of the auto-encoder with a first layer made of *slack variables*. Formally, to each feature f_i is associated a (learned) coefficient a_i in [0,1], and the encoder is fed with the vector $(a_i f_i)$ (fig. 1). The learning criterion here is the reconstruction loss augmented with an L_1 penalization on the slack variables:

$$L(F) = \sum_{i=1}^{D} ||\hat{f}_i - f_i||_2^2 + \lambda \sum_{i=1}^{D} |a_i| \qquad (10)$$

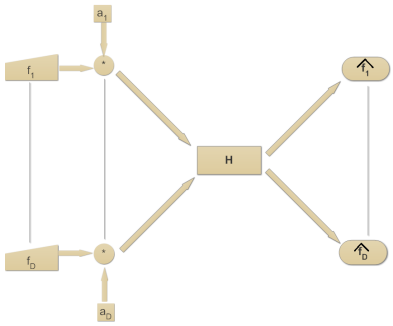


Figure 1: Structure of the neural network used in AgnoS-S

Like in LASSO, the L_1 penalization pushes the slack variables toward a sparse vector.

Eventually, the score of the i-th feature is set to $|a_i|$: this single valued coefficient reflects the contribution of f_i to the latent representation, and its importance to reconstruct the whole feature set.

Algorithm 3 AgnoS-S

Input : Feature set $F = \{f_1, ..., f_D\}$
Parameter: λ
Output : Ranking of features in F
Normalize each feature to zero mean and unit variance.
Estimate intrinsic dimension $\widehat{ID}$ of F.
Initialize neural network with $(a_1, ..., a_D) = \mathbf{1_D}$ and $d = \widehat{ID}$ neurons in the hidden layer.
Repeat
Backpropagate $L(F) = \sum_{i=1}^{D} ||\hat{f}_i - f_i||_2^2 + \lambda \sum_{i=1}^{D} |a_i|$
until convergence
Rank features by decreasing scores with $Score_S(f_i) = |a_i|$.

3 Related work

Let us first briefly present related work in unsupervised feature selection, before discussing the position of the proposed AgnoS.

Most unsupervised FS algorithms rely on *spectral clustering theory* (Luxburg, 2007). Let *sim* and M respectively denote a similarity metric on the instance space, e.g. $sim(x_i, x_j) = exp\{-\|x_i - x_j\|_2^2\}$ and M the $n \times n$ matrix with $M_{i,j} = sim(x_i, x_j)$.

Let Δ be the diagonal degree matrix associated with M, i.e. $\Delta_{ii} = \sum_{k=1}^{n} M_{ik}$, and $L = \Delta^{-\frac{1}{2}}(\Delta - M)\Delta^{-\frac{1}{2}}$ the normalized Laplacian matrix associated with M.

Spectral clustering relies on the diagonalization of L, with λ_i (resp. ξ_i) the eigenvalues (resp. eigenvectors) of L, with $\lambda_i \leq \lambda_{i+1}$. Informally, the ξ_i are used to define soft cluster indicators (i.e. the degree to which x_k belongs to the i-the cluster being proportional to $\langle x_k, \xi_j \rangle$), with λ_k measuring the inter-cluster similarity (the smaller the better).

The general unsupervised clustering scheme proceeds by clustering the samples and falling back on supervised feature selection by considering the clusters as if it were classes; more precisely, the features are assessed depending on how well they separate clusters.

Early unsupervised clustering approaches, such as the *Laplacian score* (He et al., 2005) and *SPEC* (Zhao and Liu, 2007), score each feature depending on its average alignment with the dominant eigenvectors $(\langle f_i, \xi_k \rangle)$.
A finer-grained approach is *MCFS* (Cai et al., 2010), that pays attention to the local informativeness of features and evaluates features on a per-cluster basis. Each feature is scored by its maximum alignment over the set of eigenvectors $(max_k \langle f_i, \xi_k \rangle)$.

Letting A denote the feature importance matrix, with $A_{i,k}$ the relevance score of f_i for the k-th cluster, *NDFS* (Li et al., 2012) aims to actually reduce the number of features, and jointly optimizes the cluster indicator matrix Ξ (initialized from eigenvectors $\xi_1, \ldots, \xi_n$) and the feature importance matrix A, with a sparsity constraint on the rows of A (few features should be relevant).

SOGFS (Nie et al., 2016) goes one step further: after each learning iteration on Ξ and A, M is recomputed where the distance/similarity among the samples is biased to consider only the most important features according to A.

Discussion. A first difference between the previous approaches and the proposed AgnoS, is that the spec-

tral clustering approaches (with the except of Nie et al. (2016)) rely on the Euclidean distance between points in $\mathbb{R}^D$. Due to the curse of dimensionality (Duda et al., 2012) however, the Euclidean distance in high dimensional spaces is notoriously poorly informative, with all samples being far from each other. Quite the contrary, AgnoS builds upon a non-linear dimensionality reduction approach, mapping the data onto a low-dimensional space.

Another difference regards the robustness of the approaches w.r.t. the redundancy of the initial representation of the data. Redundant features can indeed distort the distance among points, and thus bias spectral clustering methods, with the except of Li et al. (2012); Nie et al. (2016). In practice, highly correlated features tend to get similar scores according to *Laplacian score*, *SPEC* and *MCFS*. Furthermore, the higher the redundancy, the higher their score, and the more likely they will *all* be selected. This weakness is addressed by *NDFS* and *SOGFS* via the sparsity constraint on the rows of A, making it more likely that only one out of a cluster of redundant features be selected.

Finally, a main difference between the cited approaches and ours is the ultimate goal of feature selection, and the assessment of the methods. As said, unsupervised feature selection methods are assessed along a supervised setting: considering a target feature f^* (not in the feature set), the FS performance is measured from the accuracy of K-means clustering performed w.r.t. the selected features. This assessment procedure thus critically depends on the relation between f^* and the features in the feature set. Quite the contrary, the proposed approach aims to data compression; it does not ambition to predict some target feature, but rather to approximate *every* feature in the feature set.

4 Experimental setting

4.1 Goal of experiments

Our experimental objective is threefold: we aim to compare the three versions of AgnoS to unsupervised FS baselines w.r.t. both supervised evaluation and data compression. Thirdly, these experiments will serve to confirm or infirm our claim that the typical supervised evaluation scheme is unreliable.

4.2 Experimental setup

Benchmark datasets Experiments are carried on 8 datasets taken from the scikit-feature database (Li et al., 2018), an increasingly popular benchmark for feature selection (Chen et al., 2017). These datasets include face image, sound processing and medical data. In all datasets but one (Isolet), the number of samples is small w.r.t. the number of features D. Dataset size, dimensionality, number of classes, estimated intrinsic dimension[3] and data type are summarized in Table 1.

	# samples	# features	# classes	Estimated ID	Data type
arcene	200	10000	2	40	Medical
Isolet	1560	617	26	9	Sound processing
ORL	400	1024	40	6	Face image
pixraw10P	100	10000	10	4	Face image
ProstateGE	102	5966	2	23	Medical
TOX171	171	5748	4	15	Medical
warpPie10P	130	2400	10	3	Face image
Yale	165	1024	15	10	Face image

Table 1: Summary of benchmark datasets

The fact that the estimated ID is small compared to the original dimensionality for every dataset highlights the potential of feature selection for data compression.

Methods and parameters AgnoS-W, AgnoS-G and AgnoS-S are compared to 4 unsupervised FS baselines : the *Laplacian score* (He et al., 2005), *SPEC* (Zhao and Liu, 2007), *MCFS* (Cai et al., 2010) and *NDFS* (Li et al., 2012). All implementations have also been taken from the scikit-feature database, and all their hyperparameters have been set to their default values. In all experiments, the three variants of AgnoS are ran using a single hidden layer, *tanh* activation for both encoder and decoder, *Adam* (Kingma and Ba, 2015) adjustment of the learning rate, initialized to 10^{-2}. Dimension d of the hidden layer is set for each dataset to its estimated intrinsic dimension. With d fixed, preliminary experiments have shown a low sensitivity of results w.r.t. λ in the range $[10^{-1}, ..., 10^{1}]$, and degraded performance for values of λ far outside this range in either direction. Therefore, the value of λ is set to 1. The AE weights are initialized after Glorot and Bengio (2010). Each performance indicator is averaged on 10 runs with same setting; the variance is negligible.

Performance indicators For a given benchmark dataset, unsupervised FS is first performed with the 4 baseline methods and the three AgnoS variants, each algorithm producing a ranking S of the original features. Two performance indicators, one *supervised*

[3]The estimator from Facco et al. (2017) was used as this estimator is empirically less computationally expensive, requires less datapoints to be accurate, and is more resilient to high-dimensional noise than all other ID estimators introduced in section 2.2.

and one *unsupervised*, are then computed to assess and compare the different rankings.

Following the typical supervised evaluation scheme, the first indicator is the K-means *clustering accuracy (ACC)* (He et al., 2005; Cai et al., 2010) for predicting the ground truth target f^*. In the following, clustering is performed considering the top $k = 100$ ranked features w.r.t. S, with $K = c$ clusters, where c is the number of classes in f^*.

The second indicator corresponds to the unsupervised FS goal of recovering *every* initial feature f. For each $f \in F$, a 5-NearestNeighbor regressor is trained to fit f, where the neighbors of each point x are computed considering the Euclidean distance based on the top $k = 100$ ranked features w.r.t. S. The goodness-of-fit is measured via the R^2 *score* (a.k.a. *coefficient of determination*) $R^2(f, S) \in]-\infty, 1]$ [4]. The unsupervised performance of S is the individual R^2 score averaged over the *whole* feature set F:

$$Score(S) = \frac{1}{D} \sum_{j=1}^{D} R^2(f_j, S)$$

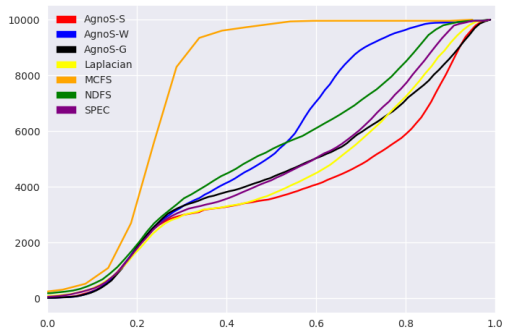


Figure 2: Cumulative distribution functions of the R^2 scores of a 5-NearestNeighbors regressor using the top 100 ranked features on *Arcene*. If a point has coordinates (x, y), then the goodness-of-fit of the regressor is $\leq x$ for y initial features.

5 Experimental results and discussion

Table 2 reports the supervised ACC scores on the ground truth labels for each selection method and benchmark dataset. On every dataset but one (*TOX171*), the highest recorded ACC is achieved by one of the three AgnoS variants, showing that AgnoS is competitive with the baselines w.r.t. *supervised* evaluation. Additionally, AgnoS-S performs better than its two counterparts AgnoS-W and AgnoS-G on average. We will now examine the *unsupervised* results.

Figure 2 depicts the respective cumulative distribution functions of the R^2 scores achieved by a 5-NearestNeighbors regressor using the top 100 ranked features by the baseline methods and the three AgnoS variants, on *Arcene*. A first observation is that every FS algorithm leads to accurate fitting (R^2 score > 0.8) of *some* features and poor fitting (R^2 score < 0.2) on *some* other features. This shows that the quality of the model predictions is very sensitive w.r.t. the target variable, which is a first supportive argument to our claim that supervised assessment of unsupervised FS (dealing with a *single* target) is unreliable.

Another interesting result is that FS algorithms differ in the number of poorly fitted features. R^2 scores < 0.2 are achieved for less than 20% of features using any declination of AgnoS and more than 35% of features using MCFS. This shows that on this example dataset, AgnoS retains information about more features than MCFS.

Table 3 contains the average R^2 score of a 5-NearestNeighbors regressor on the whole feature set, for each FS algorithm and dataset. AgnoS-S is shown to achieve a higher mean R^2 score than AgnoS-W, AgnoS-G and the baselines on all datasets. These results empirically demonstrate that the selection subsets induced by AgnoS-S retain more information about the features *on average* than the baselines.

Table 4 contains the respective frequencies of ranks attained by each selection method w.r.t. the R^2 scores of each feature on the *warpPIE10P* dataset. A first observation is that every FS algorithm is able to achieve any rank, depending on the target variable. Therefore, comparison of baseline methods using a single ground truth target, as is the case in typical supervised assessment, is unreliable. We also notice that not only is AgnoS-S more often ranked first than the baselines, it is also least often ranked last. This property also holds true for AgnoS-W and AgnoS-G, although the contrast with the baselines is less pronounced.

Throughout these experiments, AgnoS-W and AgnoS-G have been shown to perform worse than

[4] Score value $R^2(f, S) = 1$ indicates that the regressor is able to perfectly fit f. $R^2(f, S) \leq 0$ indicates that the regressor is worse than the trivial model outputting the average value of f no matter the input x.

	Arcene	Isolet	ORL	pixraw10P	ProstateGE	TOX171	warpPIE10P	Yale
AgnoS-S	**0.665**	0.536	**0.570**	**0.812**	**0.608**	0.404	0.271	0.509
AgnoS-W	0.615	**0.583**	0.548	0.640	0.588	0.292	0.358	0.382
AgnoS-G	0.630	0.410	0.528	0.776	0.569	0.357	**0.419**	**0.533**
Laplacian	0.660	0.482	0.550	0.801	0.578	0.450	0.295	0.442
MCFS	0.550	0.410	0.562	0.754	0.588	**0.480**	0.362	0.400
NDFS	0.510	0.562	0.538	0.783	0.569	0.456	0.286	0.442
SPEC	0.655	0.565	0.468	0.482	0.588	0.474	0.333	0.400

Table 2: Clustering ACC score on the ground truth labels, using the top 100 ranked features. Statistically significantly (according to a t-test with a p-value of 0.05) better results in boldface.

	Arcene	Isolet	ORL	pixraw10P	ProstateGE	TOX171	warpPIE10P	Yale
AgnoS-S	**0.610**	**0.763**	**0.800**	**0.855**	**0.662**	**0.581**	**0.910**	**0.703**
AgnoS-W	0.460	0.762	0.795	0.782	0.620	0.580	0.897	0.696
AgnoS-G	0.560	0.701	0.780	0.832	0.606	0.528	0.901	0.671
Laplacian	0.576	0.680	0.789	0.840	0.655	0.563	0.903	0.601
MCFS	0.275	0.720	0.763	0.785	0.634	0.549	0.870	0.652
NDFS	0.490	0.747	0.796	0.835	0.614	0.520	0.904	0.677
SPEC	0.548	0.733	0.769	0.761	0.646	0.559	0.895	0.659

Table 3: Average of R^2 score of 5-NearestNeighbors regressor fitting *any* feature, using the top 100 ranked features. Statistically significantly (according to a t-test with a p-value of 0.05) better results in boldface.

	1	2	3	4	5	6	7
AgnoS-S	0.37	0.09	0.12	0.09	0.17	0.07	0.08
AgnoS-W	0.16	0.11	0.09	0.23	0.07	0.21	0.13
AgnoS-G	0.12	0.17	0.11	0.18	0.05	0.24	0.13
LAP	0.12	0.17	0.20	0.17	0.04	0.17	0.13
MCFS	0.10	0.23	0.11	0.19	0.04	0.16	0.17
NDFS	0.08	0.21	0.13	0.11	0.11	0.09	0.27
SPEC	0.04	0.03	0.25	0.03	0.52	0.05	0.09

Table 4: Frequency of ranks of selection methods w.r.t. R^2 scores of each feature on warpPIE10P with a 5-NearestNeighbors regressor using the top 100 ranked features

AgnoS-S (and even the baselines). Our interpretation is that this is due to a key difference between the LASSO regularization and the slack variables. On one hand, the encoder weights in AgnoS-W (resp. the encoder gradients in AgnoS-G) are simultaneously responsible for producing the compressed data representation and enforcing sparsity among the original features. These parameters therefore play two potentially conflicting roles.

On the other hand, the slack variables in AgnoS-S are only subject to the sparsity pressure exerted by the L_1 penalty ; these coefficients do not directly take part in data compression, and are hardly influenced by the MSE part of the loss. Therefore, we hypothesize that this allows the slack variable layer to enforce sparse feature selection more efficiently than AgnoS-W and AgnoS-G.

A sensitivity analysis of the approach w.r.t. the number of selected features has been conducted and

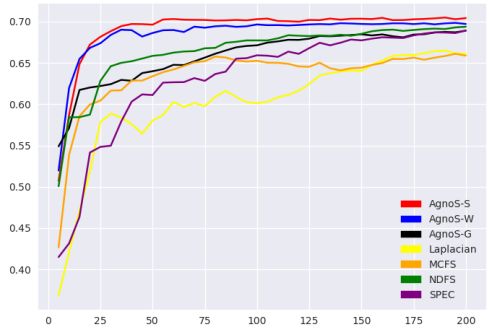


Figure 3: Average R^2 score on *Yale* w.r.t. the number k of top ranked features considered

is summarized on Figure 3, reporting the R^2 score (averaged on the whole feature set) achieved by a 5-NearestNeighbors regressor on the *Yale* dataset for $k \in [5, 10, \ldots, 200]$. AgnoS-S is shown to reliably outperform the baselines for every value of k (with the exception of $k \in \{5, 10\}$ where it is tied with NDFS).

Additionally, the *unsupervised* ranking of the considered FS algorithms appears to be stable [5] w.r.t. the selection subset size. This stability property does not hold using the ACC score, for which additional experiments have shown that the *supervised* ranking of FS

[5] The rank of each method changes at most 3 times with k, and the top ranked algorithms are mostly invariant across the range $[5, 10, \ldots, 200]$.

algorithms is sensitive w.r.t. k (Doquet, 2019). This is an additional empirical argument against the typical supervised evaluation scheme.

	arcene	Isolet	ORL	pixraw10P	ProstateGE	TOX171	warpPie10P	Yale
AgnoS	265	25	29	242	145	143	31	14
Laplacian	<1	<1	<1	<1	<1	<1	<1	<1
SPEC	3	9	<1	2	1	2	1	<1
MCFS	<1	2	<1	<1	<1	<1	<1	<1
NDFS	130	16	17	193	80	76	18	7

Table 5: Empirical runtimes on a single Nvidia Geforce GTX 1060 GPU, in seconds.

Table 5 contains the empirical runtimes of the baselines and AgnoS on each dataset, on a single Nvidia Geforce GTX 1060 GPU. AgnoS is shown to be between 25% and 100% slower than NDFS, and several orders of magnitude slower than Laplacian score, SPEC and NDFS. The computational effort thus is a limitation of the proposed approach.

6 Conclusion

In this paper, we have introduced a novel unsupervised FS algorithm based on data compression. A main merit of the proposed AgnoS-S is to better recover the whole feature set − and the target feature − compared to the baselines, in counterpart for its higher computational cost. We have also empirically shown the brittleness of the mainstream supervised assessment of unsupervised FS methods.

This work opens two perspectives for further studies. The first one is concerned with early stopping of the AE, aimed to reduce the computational cost of AgnoS. Another direction is to consider Variational AutoEncoders (VAE) (Kingma and Welling, 2013) instead of plain AEs, likewise augmenting the VAE loss with an L_1 penalization to achieve feature selection; the expected advantage of VAEs would be to be more robust when considering small datasets.

Acknowledgments

We wish to thank Diviyan Kalainathan for many enjoyable discussions. We also thank the anonymous reviewers, whose comments helped to improve the experimental setting and the assessment of the method.

References

H. Alemu, W. Wu, and J. Zhao. Feedforward neural networks with a hidden layer regularization method. *Symmetry*, 10(10), 2018.

D. Cai, C. Zhang, and X. He. Unsupervised feature selection for multi-cluster data. *International conference on Knowledge Discovery and Data mining*, pages 333–342, 2010.

F. Camastra and A. Staiano. Intrinsic dimension estimation: Advances and open problems. *Information Sciences*, 328:26–41, 2016.

G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers Electrical Engineering*, 40(1):16–28, 2014.

J. Chen, M. Stern, M. J. Wainwright, and M. I. Jordan. Kernel feature selection via conditional covariance minimization. *In Advances in Neural Information Processing Systems*, pages 6946–6955, 2017.

T. F. Cox and M. A. Cox. *Multidimensional scaling.* Chapman and hall/CRC, 2000.

G. Doquet. *Unsupervised Feature Selection.* PhD thesis, Université Paris-Sud, 2019. Published later in 2019.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv:1702.08608*, 2017.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification.* John Wiley and Sons, 2012.

E. Facco, M. d'Errico, A. Rodriguez, and A. Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Nature*, 7(1), 2017.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *International conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015.

X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. *Advances in Neural Information Processing Systems*, pages 507–514, 2005.

S. Ivanoff, F .Picard, and V. Rivoirard. Adaptive lasso and group-lasso for functional poisson regression. *The Journal of Machine Learning Research*, 17(1):1903–1948, 2016.

D. Kalainathan, O. Goudet, I. Guyon, D. Lopez-Paz, and M. Sebag. Sam: Structural agnostic model, causal discovery and penalized adversarial learning. *arXiv:1803.04929*, 2018.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

P. Leray and P. Gallinari. Feature selection with neural networks. *Behaviormetrika*, 26(1):145–166, 1999.

E. Levina and P. J. Bickel. Maximum likelihood estimation of intrinsic dimension. *In Advances in Neural Information Processing Systems*, pages 777–784, 2005.

J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6), 2018.

Z. Li, Y. Yang, Y. Liu, X. Zhou, and H. Lu. Unsupervised feature selection using non-negative spectral analysis. *AAAI*, 2012.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

L. V. D. Maaten and G. Hinton. Visualizing data using t-sne. *The Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv:1802.03426*, 2018.

L. Meier, S. Van De Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

A. Y. Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. *International Conference on Machine Learning*, 2004.

F. Nie, W. Zhu, and X. Li. Unsupervised feature selection with structured graph optimization. *AAAI*, pages 1302–1308, 2016.

S. Sadeghyan. A new robust feature selection method using variance-based sensitivity analysis. *arXiv:1804.05092*, 2018.

L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning research*, 4(Jun):119–155, 2003.

N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

L. Song, A. Smola, A. Gretton, J. Bedo, and K. Borgwardt. Feature selection via dependence maximization. *Journal of Machine Learning Research*, 13: 1393–1434, 2012.

J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B (Methodological)*, pages 267–288, 1996.

D. Varga, A. Csiszárik, and Z. Zombori. Gradient regularization improves accuracy of discriminative models. *arXiv:1712.09936*, 2017.

T. Wiatowski and H. Bölcskei. A mathematical theory of deep convolutional neural networks for feature extraction. *IEEE Transactions on Information Theory*, 64(3):1845–1866, 2018.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2007.

Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. *International Conference on Machine Learning*, 2007.

Classification ascendante hiérarchique, contrainte d'ordre : conditions d'applicabilité, interprétabilité des dendrogrammes

Nathanaël Randriamihamison[1,2], Pierre Neuvial[2], et Nathalie Vialaneix[1]

[1]MIAT, Université de Toulouse, INRA, Castanet Tolosan, France
[2]Institut de Mathématiques de Toulouse, UMR 5219, Univ. de Toulouse, CNRS, France

9 avril 2019

Résumé

La classification ascendante hiérarchique (CAH) avec lien de Ward est une approche fréquemment utilisée pour partitionner des objets d'un espace multi-dimensionnel. Nous discutons ici de la validité théorique des extensions de cette approches à différents types d'entrées (distances, dissimilarités, noyaux, similarités) ainsi que de la pertinence de l'utilisation de sa version contrainte (parfois appelée « CONISS ») dans ces divers cas. En particulier, nous étudions les conditions garantissant la cohérence entre les résultats de la CAH et leur représentation graphique classique sous forme de dendrogramme. Nous présentons également une étude de simulations montrant que la version contrainte de la méthode permet, outre des gains computationnels, d'obtenir de meilleurs résultats de classification lorsque la contrainte est cohérente avec la structure des données. Beaucoup de ces résultats sont présents dans une littérature hétéroclite : l'objet de cette communication est de proposer donc un cadre de présentation uniforme et de compléter les résultats existants.

Mots-clefs : Classification ascendante hiérarchique, lien de Ward, classification ascendante hiérarchique sous contrainte, dendrogramme, croissance.

1 Classification Ascendante Hiérarchique (CAH)

Cadre euclidien standard

Dans le cadre standard de la Classification Ascendante Hiérarchique (CAH), on suppose que l'ensemble des objets $\Omega = \{x_1, \ldots, x_n\}$ est un sous-ensemble de $\mathbb{R}^p$, et que la relation de proximité entre ces objets est encodée par la distance $D = (d_{ij})_{1 \leq i,j \leq n}$ induite par la norme euclidienne de $\mathbb{R}^p$ avec $d_{ij} = \|x_i - x_j\|_{\mathbb{R}^p}$.

L'algorithme de CAH part de la partition triviale en singletons $\mathcal{P}_1$, puis par fusions successives, se termine sur une autre partition triviale, $\mathcal{P}_n = \Omega$. L'étape t, permettant de passer de la partition $\mathcal{P}_t$ à la partition $\mathcal{P}_{t+1}$, fusionne les deux classes de $\mathcal{P}_t$ les plus proches au sens d'une dissimilarité entre classes appelée *critère de lien* et notée δ.

Le *lien de Ward* [War63] est fréquemment utilisé car il a une interprétation simple. Pour un ensemble quelconque $G \subset \Omega$, l'inertie de G est définie par $I(G) = \sum_{x \in G} \|x - \bar{x}_G\|^2$, où $\bar{x}_G = |G|^{-1} \sum_{x \in G} x$ est le centre de gravité de G, et $|G|$ le cardinal de G. On appelle inertie intra-classes d'une partition $\mathcal{P} = (G_1, G_2, ..., G_K)$ la somme des inerties des classes : $I(\mathcal{P}) = \sum_{k=1}^{K} I(G_k)$. Le lien de Ward entre deux sous-ensembles disjoints G et G' est défini par : $\delta(G, G') = I(G \cup G') - I(G) - I(G')$ et correspond alors à la variation d'inertie intra-classes induite par cette fusion.

Extensions de la CAH

La CAH est couramment utilisée sur des données plus générales que celles décrites dans le cadre euclidien de la section précédente. On suppose désormais que les objets $(x_i)_i$ appartiennent à un espace arbitraire (non nécessairement euclidien).

Dissimilarités. Considérons tout d'abord le cas où les objets sont décrits par une mesure de dissimilarité entre paires, $d_{ij} = d(x_i, x_j)$, qui n'est ici plus nécessairement la distance euclidienne. On suppose alors que la matrice $D = (d_{ij})_{1 \leq i,j \leq n}$ est une matrice symétrique à coefficients positifs et à diagonale nulle. Remarquons que ce cas est une extension du

cas euclidien décrit dans la section précédente, dans lequel l'inertie $I(G)$ s'exprime directement à partir des éléments de la matrice de distance D comme $I(G) = (2|G|)^{-1} \sum_{(x_i,x_j)\in G^2} d_{ij}^2$. On peut alors définir, par analogie au cas euclidien, une extension de l'inertie et du lien de Ward associée à D (voir par exemple [SR05] ou [CKSLS18]). Cependant, dans ce cas, on perd l'interprétabilité en termes de centre de gravité.

Noyaux. Dans un certain nombre de cas pratiques, les objets sont décrits par leurs ressemblances et non leurs dissemblances. C'est le cas, en particulier, lorsque les données sont décrites par un noyau [STV04], c'est-à-dire, par une matrice symétrique définie positive $K = (k(x_i, x_j))_{1\leq i,j\leq n}$. [Aro50] montre que le noyau peut être interprété comme une matrice de produit scalaire dans un certain espace de représentation $\mathcal{H}$. Ce cadre-ci est donc équivalent au cadre euclidien et on peut montrer que l'expression du lien de Ward s'écrit à partir des valeurs du noyau uniquement en utilisant l'*astuce noyau* [Deh15] :

$$\delta(G, G') = \frac{K(G)}{|G|} + \frac{K(G')}{|G'|} - \frac{K(G \cup G')}{|G \cup C'|}, \quad (1)$$

avec $K(G) = \sum_{x_i,x_j\in G^2} k(x_i, x_j)$ pour un sous-ensemble, G, d'éléments de $(x_i)_i$.

Similarités. On peut interpréter la notion de similarité comme une généralisation de celle de noyau. Bien qu'il n'y ait pas de consensus sur la définition exacte d'une similarité, on appellera similarité une matrice $S = (s_{ij})_{1\leq i,j\leq n}$ symétrique à diagonale positive. [MAET15] ont montré que l'approche « noyau » décrite ci-dessus pouvait être généralisée à toute matrice de similarité, par l'argument suivant : considérons la matrice $S + \lambda I_n$, c'est-à-dire la matrice S dont les éléments diagonaux sont translatés de λ. Alors, d'une part, pour λ suffisamment grand, S_λ est définie positive, et peut donc s'interpréter comme un noyau. D'autre part, appliquer (formellement) l'algorithme de CAH à la matrice de similarité S_λ en utilisant l'équation (1) avec $K = S_\lambda$ aboutit exactement à la même suite de fusions, quelle que soit la valeur de λ. En effet, l'équation (1) implique : $\delta_{S_\lambda} = \delta_{S_{\lambda'}} + (\lambda - \lambda')$.

Dans la suite, nous distinguerons simplement deux cas : le cas euclidien (qui contient d'après ce qui précède les cas « noyau » et « similarité quelconque »), et le cas non-euclidien, qui correspond aux dissimilarités quelconques.

CAH sous contrainte d'ordre

Dans un certain nombre de contextes applicatifs, il existe une information *a priori* sur les relations entre les objets. C'est le cas, en particulier, en statistique spatiale où les objets spatiaux sont en relation de voisinage, ou bien dans le contexte génomique, où les loci (positions) génomiques sont ordonnés le long d'une ligne (le chromosome, voir [ADN+19] pour des applications aux données de génétique, SNP, et aux données de conformation spatiale de la chromatine, Hi-C). La classification ascendante hiérarchique sous contrainte de contiguïté [FB82] restreint les fusions possibles aux objets dits contigus.

Dans la suite, on s'intéresse au cas particulier de la Classification Ascendante Hiérarchique sous Contrainte d'Ordre (CAHCO) : les objets initiaux sont reliés par une relation d'ordre total (temps, position génomique), et deux classes sont dites contiguës si et seulement si on peut en extraire un couple d'objets contigus. Cette version est souvent appelée « CONISS » pour CONstrained Incremental Sum of Squares [Gri87] lorsque le lien de Ward est utilisé pour la fusion des classes.

L'intérêt de l'ajout d'une contrainte est double : d'une part, elle permet de répercuter au mieux les relations existantes entre les objets au cours de la procédure, et ainsi de rendre les résultats plus interprétables. La CAHCO peut ainsi être vue comme une méthode de segmentation de données ordonnées linéairement. D'autre part, la complexité en temps de la CAH sans contrainte est cubique ($O(n^3)$) et celle de la CAHCO est seulement quadratique ($O(n^2)$) [Deh15]. Lorsque les données d'entrées sont une similarité creuse dont les éléments non nuls sont proches de la diagonale, [ADN+19] ont proposé une approche permettant d'obtenir les résultats d'une CAHCO avec une complexité quasi-linéaire. Dans ce cas particulier, la CAHCO peut être utilisée comme une heuristique intéressante pour la segmentation.

2 Dendrogrammes

Les résultats d'une CAH sont fréquemment représentés à l'aide d'un *dendrogramme*, comme sur la figure 1. Un dendrogramme est un arbre binaire dont les nœuds sont les classes de la hiérarchie. Dans le cas particulier de la CAHCO, les feuilles, qui correspondent aux n objets à classer, sont représentées selon l'ordre naturel de ces objets. La hauteur du nœud correspondant à la t-ème fusion est notée h_t (les feuilles sont à la hauteur $h_0 = 0$). On utilise souvent comme hauteur la valeur du lien de Ward à l'étape t, notée m_t.

Une propriété naturelle attendue pour le dendro-

gramme d'une CAH est que la suite de partitions induite par la CAH coïncide avec celle décrite par le dendrogramme. Cette propriété est équivalente à la croissance de la suite $(h_t)_t$. En effet, lorsqu'un dendrogramme est construit avec une suite de hauteurs non croissante (comme sur la figure 1 entre les fusions 3 et 4), il est difficilement interprétable et l'obtention d'une partition par coupe du dendrogramme a une hauteur donnée n'est pas définie correctement et n'est pas cohérente avec la suite de partitions fournies par la CAH. En particulier, lorsque la hauteur est définie par le lien de Ward, la croissance n'est pas nécessairement vérifiée pour la CAHCO [Gri87].

[Gri87] décrit des choix de hauteurs alternatifs au lien de Ward pour pallier ce problème. Parmi ceux-ci, on trouve **l'inertie intra-classes**, qui est fréquemment utilisée dans le cadre de la CAHCO (c'est notamment cette hauteur qui est fournie par la version de la CAHCO implémentée dans le package R **rioja**). Pour ce critère, la hauteur à l'étape t est définie par $\mathrm{ESS}_t = \sum_{u=1}^{n-t} I(G_u^{t+1})$. Comme pour m_t, cette hauteur est croissante pour la CAH [War63, Bat81]. Sa croissance pour la CAHCO n'est pas non plus assurée dans le cas non-euclidien, mais elle l'est dans le cas euclidien [Gri87]. La figure 1 donne un exemple simple où la CAHCO est appliquée à six objets décrits par la dissimilarité $D = \begin{pmatrix} 0 & \sqrt{1.99} & \sqrt{1.99} & \sqrt{1.99} & 0.1 & 1 \\ \sqrt{1.99} & 0 & \sqrt{2} & \sqrt{1.99} & 0.1 & 1 \\ \sqrt{1.99} & \sqrt{2} & 0 & \sqrt{2} & 0.1 & 1 \\ \sqrt{1.99} & \sqrt{1.99} & \sqrt{2} & 0 & \sqrt{2} & 1 \\ 0.1 & 0.1 & 0.1 & \sqrt{2} & 0 & \sqrt{2} \\ 1 & 1 & 1 & 1 & \sqrt{2} & 0 \end{pmatrix}$, pour lesquels on obtient des hauteurs non croissantes : $h_1 < h_2 < h_4 < h_3 < h_5$.

3 Comparaison de CAH et CAHCO

La CAH est souvent perçue comme une approche gloutonne permettant de déterminer une partition des données avec une inertie intra-classes, ESS_t, minimale pour un nombre donné, k ($\in \{1, \ldots, n\}$) de classes. Aussi, il est raisonnable d'attendre que l'inertie des partitions obtenues avec la CAH standard soit inférieure à l'inertie des partitions obtenues par une CAH contrainte comme la CAHCO, dans laquelle l'ensemble des fusions possibles est restreint. Dans cette section, nous montrons que dans le cas où les données sont structurées de manière « compatible » avec la contrainte imposée à la CAHCO, non seulement l'algorithme contraint est plus efficace que l'algorithme standard (en termes de complexité en temps et espace), mais il permet aussi d'obtenir des partitions de plus faible inertie intra-classes, c'est-à-dire meilleures. Pour ce faire, nous proposons un processus aléatoire de perturbation de la structure originale de données génomiques et quantifions l'impact de cette perturbation sur la qualité des résultats des classifications.

Données et méthode de simulation

Pour cela, nous avons analysé des données Hi-C [DSY+12], qui présentent une forte structure d'ordre. Les données Hi-C permettent d'étudier les relations de proximité dans l'espace (3D) de la chromatine en mesurant la fréquence d'interactions physiques entre paires de positions génomiques par du séquençage haut-débit. De manière formelle, les données Hi-C peuvent être représentées sous la forme d'une matrice symétrique, $S = (s_{ij})_{i,j}$ dans laquelle chaque valeur s_{ij} correspond au nombre d'interactions mesurées entre les positions génomiques i et j. Les positions génomiques i et j représentent des intervalles de longueur identique le long de la séquence d'ADN, et les valeurs s_{ij} sont par définition des entiers positifs (ou nuls). La matrice présente une structure diagonale forte, qui reflète l'organisation linéaire le long des chromosomes, comme illustré dans la figure 2 qui représente la partie triangulaire supérieure d'une carte Hi-C. Comme décrit dans [ADN+19], la classification contrainte des positions génomiques selon cette matrice de similarité est liée à des questions biologiques de compréhension de la structure de la chromatine (détection de TADs [ZTOC18] ou de compartiments actifs/inactifs [LAvBW+09]).

Dans les simulations qui suivent, nous avons utilisé un chromosome (le chromosome 3) d'une expérience sur des cellules souches publiée dans [DSY+12] [1]. Les données ont été log-transformées avant les classifications, pour limiter l'asymétrie des valeurs ; la correction de la similarité décrite dans la section 1 pour transformer cette similarité en noyau a également été utilisée.

Pour quantifier l'influence de la structure des données sur la qualité des partitions obtenues, nous avons utilisé un processus de perturbation des données initiales qui gomme progressivement la structure diagonale. Ce processus consiste à échanger, de manière aléatoire, des paires d'entrées, s_{ij} et $s_{i'j'}$, où (i, j) et (i', j') sont tirés de manière uniforme parmi l'ensemble des paires (u, v), $(u', v') \in \{1, \ldots, n\}$ telles que

1. La carte Hi-C utilisée est la carte des données normalisée, disponible sur le site internet des auteurs : `http://chromosome.sdsc.edu/mouse/hi-c/download.html`.

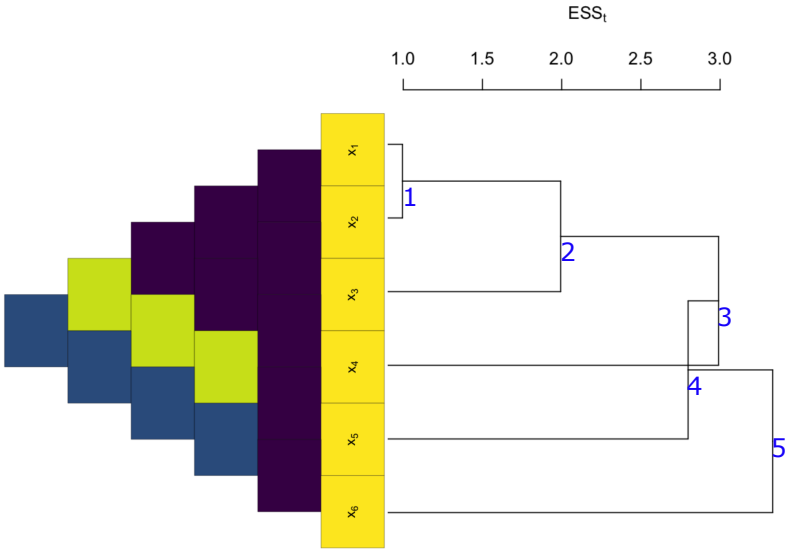


FIGURE 1 – **Un croisement dû à la non-croissance de la hauteur définie par l'inertie intra-classes, ESS$_t$, pour la CAHCO avec données non euclidiennes**. À gauche : représentation de la dissimilarité associée (les couleurs sombres correspondent à de grandes valeurs donc à des objets distants). À droite : dendrogramme correspondant aux résultats de la CAHCO (l'ordre des fusions est noté en bleu).

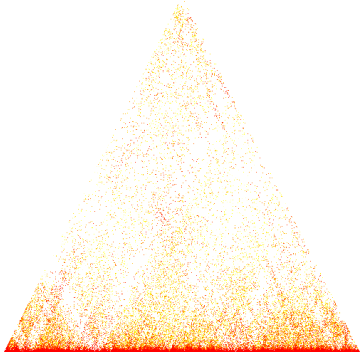


FIGURE 2 – **Partie triangulaire supérieure d'une carte Hi-C**. L'axe horizontal donne la position sur le chromosome et les niveaux de rouge indiquent l'intensité de la valeur s_{ij}.

$s_{uv} > 0$ (pour assurer de ne pas échanger deux valeurs nulles). La proportion des entrées de la matrice perturbée par rapport aux entrées de la matrice initiale varie de 1 à 30% [2]. Ce processus de simulation est répété 50 fois pour évaluer la variabilité des résultats. Tous les résultats ont été obtenus avec R [R C19]. Les résultats de la CAH standard ont été obtenus avec la fonction `hclust` (du package **stats**) et les résultats de la CAHCO ont été obtenus avec la fonction `adjClust` (du package **adjclust**).

Résultats

La figure 3 montre l'évolution de m_t (normalisée par sa valeur maximale pour la simulation) et de ESS$_t$ (normalisée par l'inertie totale, ESS$_n$) pour la CAH et la CAHCO lorsque le pourcentage de perturbation de la matrice initiale croît.

Pour les données originales (non perturbées), dans lesquelles l'organisation est cohérente avec la contrainte linéaire d'ordre, les hauteurs de la CAH standard et de la CAHCO sont très similaires. De manière

2. À cause du grand nombre de 0 dans les entrées des matrices Hi-C, 30% correspond approximativement à la proportion maximale.

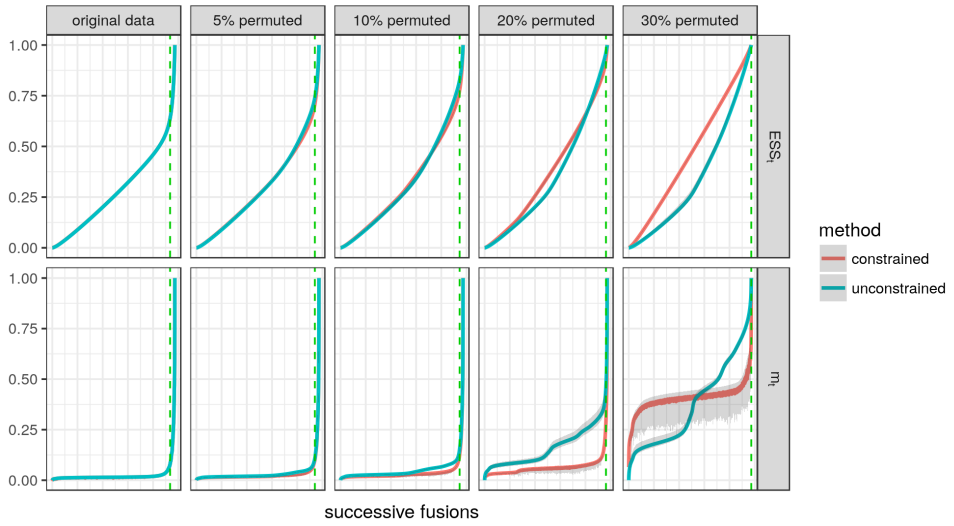


FIGURE 3 – **Comparaison de la séquence de hauteurs pour OCHAC (bleu) et HAC standard (rouge)** pour m_t (en haut) et ESS_t (en bas) pour différents niveaux de perturbation de la matrice Hi-C initiale. Les courbes correspondent à la moyenne du critère pour 50 simulations et les ombres grises au minimum et au maximum du critère sur les 50 simulations. La ligne verticale correspond au nombre de classes choisi par l'heuristique « *broken stick* ».

intéressante, la CAHCO améliore le critère ESS_t pour de faibles pourcentages de perturbations (de 5 à 10%), ce qui montre une meilleure robustesse de l'approche aux petites perturbations lorsque les données originales sont bruitées. Notons, en outre, que l'utilisation d'un critère classique de choix du nombre de classes, comme le critère « *broken stick* » [Ben96] induit la coupe du dendrogramme dans la zone où précisément ESS_t pour la CAHCO est inférieur à ESS_t pour la CAH standard. Cette propriété est intéressante pour les données rencontrées en biologie par exemple, qui présentent de nombreux biais et bruits (effets expérimentaux, biais de GC par exemple). La CAHCO est donc à privilégier dans les contextes où la contrainte d'ordre est pertinente. Enfin, pour des pourcentages élevés de perturbations (supérieurs à 20%), la CAH standard a, de nouveau, une valeur de ESS_t partout inférieure à celle obtenue avec la CAHCO. Ceci s'explique par le fait que la contrainte d'ordre n'est plus en accord, à ce niveau de perturbations, avec la structure des données. Dans ce cas, les résultats de la CAHCO ne sont plus pertinents.

Enfin, la comparaison des structures des dendrogrammes ou des classifications induites par la coupe du dendrogramme à une hauteur donnée (non montré faute de place) conduit aux mêmes observations. Les résultats sont très similaires pour CAH et CAHCO pour des données dont la structure est cohérente avec la contrainte mais deviennent rapidement différents lorsque la matrice initiale est perturbée. En particulier, l'indice γ de Baker [Bak74], permettant de comparer deux dendrogrammes, décroît très fortement à partir de 10% de perturbation et devient quasiment nul au delà de 20% de perturbation.

4 Conclusion

La CAH ainsi que sa version sous contrainte d'ordre peuvent s'appliquer de façon justifiée dans un cadre plus vaste que le cadre euclidien. En revanche, certaines propriétés théoriques qui garantissent la cohérence entre les résultats de la méthode et sa représentation graphique ne sont pas toujours garanties, en particulier, lorsque les données ne sont pas euclidiennes. Par ailleurs, nos simulations montrent que la CAH sous contrainte d'ordre peut donner des résultats de meilleure qualité que la CAH standard lorsque les données présentent une structure cohérente avec la contrainte. La complexité de la version contrainte est également inférieure à la version standard, ce qui en fait une approche pertinente pour la classification non supervisée.

Remerciements

Les auteurs remercient Marie Chavent pour des discussions stimulantes autour de ce travail. Celui-ci a été effectué dans le cadre du projet SCALES, financé par la Mission pour les Initiatives Transverses et Interdisciplinaires du CNRS. La thèse de N.R. est financée par le programme INRA/Inria.

Références

[ADN+19] Christophe Ambroise, Alia Dehman, Pierre Neuvial, Guillem Rigaill, and Nathalie Vialaneix. Adjacency-constrained hierarchical clustering of a band similarity matrix with application to genomics. arXiv preprint arXiv :1902.01596, 2019.

[Aro50] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3) :337–337, 1950.

[Bak74] Frank B. Baker. Stability of two hierarchical grouping techniques case I : sensitivity to data errors. *Journal of the American Statistical Association*, 69(346) :440–445, 1974.

[Bat81] Vladimir Batagelj. Note on ultrametric hierarchical clustering algorithms. *Psychometrika*, 46(3) :351–352, 1981.

[Ben96] Keith D. Bennett. Determination of the number of zones in a biostratigraphical sequence. *New Phytologist*, 132(1) :155–170, 1996.

[CKSLS18] Marie Chavent, Vanessa Kuentz-Simonet, Amaury Labenne, and Jérôme Saracco. ClustGeo2 : an R package for hierarchical clustering with spatial constraints. *Computational Statistics*, 33(4) :1799–1822, 2018.

[Deh15] Alia Dehman. *Spatial Clustering of Linkage Disequilibrium Blocks for Genome-Wide Association Studies*. PhD thesis, Université Paris Saclay, 2015.

[DSY+12] J.R. Dixon, S. Selvaraj, F. Yue, A. Kim, Y. Li, Y. Shen, M. Hu, J.S. Liu, and B. Ren. Topological domains in mammalian genomes identified by analysis of chromatin interactions. *Nature*, 485 :376–380, 2012.

[FB82] Anuška Ferligoj and Vladimir Batagelj. Clustering with relational constraint. *Psychometrika*, 47(4) :413–426, 1982.

[Gri87] Eric C. Grimm. CONISS : a FORTRAN 77 program for stratigraphically constrained analysis by the method of incremental sum of squares. *Computers & Geosciences*, 13(1) :13–35, 1987.

[LAvBW+09] E. Lieberman-Aiden, N.L. van Berkum, L. Williams, M. Imakaev, T. Ragoczy, A. Telling, I. Amit, B.R. Lajoie, P.J. Sabo, M.O. Dorschner, R. Sandstrom, B. Bernstein, M.A. Bender, M. Groudine, A. Gnirke, J. Stamatoyannopoulos, L.A. Mirny, E.S. Lander, and J. Dekker. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, 326(5950) :289–293, 2009.

[MAET15] Sadaaki Miyamoto, Ryosuke Abe, Yasunori Endo, and Jun-Ichi Takeshita. Ward method of hierarchical clustering for non-Euclidean similarity measures. In *Proceedings of the VIIth International Conference of Soft Computing and Pattern Recognition (SoCPaR 2015)*, Fukuoka, Japan, 2015. IEEE.

[R C19] R Core Team. *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019.

[SR05] Gábor J. Székely and Maria L. Rizzo. Hierarchical clustering via joint between-within distances : extending Ward's minimum variance method. *Journal of Classification*, 22(2) :151–183, 2005.

[STV04] Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert. *Kernel Methods in Computational Biology*. MIT PR, 2004.

[War63] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301) :236–244, 1963.

[ZTOC18] Marie Zufferey, Daniele Tavernari, Elisa Oricchio, and Giovanni Ciriello. Comparison of computational methods for the identification of topologically associating domains. *Genome biology*, 19(1) :217, 2018.

Minimisation rapide de la $\mathcal{C}$-borne avec des guaranties de généralisation.

Bauvin Baptiste*[1,2], Jean-Francis Roy[3], Cécile Capponi[1], et François Laviolette[2,4]

[1]Université Aix-Marseille, Université de Toulon, CNRS, LIS(Qarma), Marseille, France,
[2]Université Laval, Département d'informatique et de génie logiciel, Québec, QC, Canada,
[3]Coveo Solutions Inc., Québec, QC, Canada,
[4]Elements AI, Montréal, QC, Canada.

30 mai 2019

Résumé

La $\mathcal{C}$-borne est une borne serrée sur le risque d'un classifieur par vote de majorité reposant sur la qualité individuelle et le désaccord des votants et donnant des garanties PAC-Bayesiennes de généralisation. MinCQ est un classifieur qui retourne une distribution dense sur un ensemble fini de votants en minimisant la $\mathcal{C}$-borne. Introduit ensuite et inspiré par le boosting, CqBoost utilise une approche de génération de colonnes pour construire une distribution optimale pour la $\mathcal{C}$-borne sur un ensemble de votants potentiellement infini. Cependant, ces deux approches présentent un temps d'apprentissage long car ils minimisent la $\mathcal{C}$-borne grâce à un programme quadratique. Or, un avantage de CqBoost est sa capacité empirique à générer des solutions parcimonieuses. Ici, nous cherchons à accélérer la minimisation de la $\mathcal{C}$-borne tout en gardant une solution parcimonieuse et précise. Nous présentons donc un classifieur efficace basé sur une optimisation locale et gloutonne de la $\mathcal{C}$-borne, s'appuyant sur le boosting. Nous prouvons théoriquement la décroissance de la $\mathcal{C}$-borne, proposons des garanties de généralisation fondées des théorèmes PAC-Bayesiens. Puis, nous évaluons expérimentalement la pertinence de l'algorithme en termes d'erreur, de parcimonie, et de rapidité, comparativement à MinCQ, CqBoost et aux méthodes ensemblistes de l'état de l'art. Dans ces expérimentations, nous observons que la précision reste comparable à CqBoost, avec bien moins de calculs, tout en restant parcimonieux.

Mots-clef PAC-Bayes Boosting Parcimonie $\mathcal{C}$-borne Borne de généralisation

*

1 Introduction

Ensemble-based supervised classification consists in training a combination of many classifiers learnt from various algorithms or/and subsamples of the initial dataset. Some ensemble-based algorithms are competing advanced learning methods. Among these approaches, let us cite Boosting with the seminal elegant Adaboost [7] which has been the inspiration of numerous other algorithms, Bagging [1] leading to the successful Random Forests [2], but also Multiple Kernel Learning (MKL) approaches [22][12] or even the Set Covering Machines [15]. Most of these approaches are founded on theoretical aspects of PAC learning [23]. Among them, the PAC-Bayesian theory studies the properties of the majority vote used to combine the voters [16]. Then, valuable ad-hoc studies have been made over specific application domains in order to build relevant set of classifiers. We address here the problem of learning a set of classifiers independently from the priors relevant to the application domain, together with a weighted schema that would underlie a majority vote over the members of that set of classifiers.

Introduced by David McAllester [16], the PAC-Bayesian theory provided among the tightest Probably Approximately correct (PAC) learning bounds. These bounds are often used to better understand the learning capability of various algorithms [20, 17, 13, 3, 21, 6]. Based on the fact that PAC-Bayesian bounds gave rise to a nice analysis of any algorithm behavior, it has incited a research direction that consists in developing new (or new variants of) algorithms that simply are bound minimizers [8, 18, 6, 9]. In this paper, we revisit one of such algorithm, MinCq [9]. This algorithm

focuses on the minimization of the $\mathcal{C}$-bound and comes with PAC-Bayesian guarantees. The $\mathcal{C}$-bound has been introduced in [10] and bounds the risk of a weighted majority vote based on the averaged risk of the voters together with the averaged pairwise disagreement of the voters. According to the $\mathcal{C}$-bound, the quality of each individual voter can be compensated if the voting community tends to balance the individual errors by having a plurality of opinion on "difficult" examples.

Although MinCq has state of the art performance on many tasks, it finds the output distribution on a set of voters through a quadratic program, which is not tractable for more than medium-sized datasets. To overcome this, CqBoost [19] has been proposed. It iteratively builds a sparse majority vote from a possibly infinite set in a column generation setting. However, CqBoost approach only partially tackles the computational challenge. To remedy to this drawback, we propose CB-Boost, a greedy, boosting-based, $\mathcal{C}$-bound minimization algorithm designed to greatly reduce the computational cost of CqBoost and MinCq while maintaining the attractive peculiarities of the $\mathcal{C}$-bound.

In this work, we introduce CB-Boost that relies on iteratively building a set of base classifiers and weights to generate a majority vote. Hence CB-Boost is similar to CqBoost, but closer to boosting in the sense that at each iteration, it selects a voter, finds the associated weight by minimizing an objective quantity (the $\mathcal{C}$-bound for CB-Boost, and the exponential loss as for Adaboost), and will not go back on its decision.

The main property of CB-Boost comes from the fact that at each iteration it solves a $\mathcal{C}$-bound minimization problem by considering only two classifiers : the majority vote built at the previous iteration and the voter to add at the current iteration. Interestingly, it is possible to solve this minimization problem analytically and with very few operations. Furthermore, we derive from this solution a guarantee that the $\mathcal{C}$-bound decreases throughout CB-Boost's iterations.

After setting up basic notations and definitions, followed by a quick review about the $\mathcal{C}$-bound and its PAC-Bayesian framework, two algorithms aimed at learning an ensemble of classifiers are sketched upon, based on the minimization of the $\mathcal{C}$-bound : MinCq and CqBoost. From the advantages and drawbacks of these two major algorithms, this paper focuses on a new boosting-based algorithm CB-Boost (Section 3) which retains the benefits while reducing the disadvantages. Section 4 addresses the theoretical properties of CB-Boost, then Section 5 focuses on experiments that not only validate the theoretical aspects, but also demonstrate that CB-Boost performs well empirically.

2 The PAC-Bayes context

After basic definitions under notations set out below, the context of PAC-Bayes learning is introduced through the $\mathcal{C}$-bound and a couple of theorems about it, which are pivotal components of our work.

2.1 Basic notations and definitions

Let us consider a supervised biclass learning classification task, where $\mathcal{X}$ is the input space and $\mathcal{Y} = \{-1, 1\}$ is the output space. The learning sample $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m$ consists of m examples drawn *i.i.d* form a fixed but unknown distribution D over $\mathcal{X} \times \mathcal{Y}$. Let $\mathcal{H}$ be a set of n voters $h : \mathcal{X} \to \{-1, 1\}$.

Definition 1. *$\forall x \in \mathcal{X}$, the majority vote classifier (Bayes classifier) B_Q over a posterior distribution Q on $\mathcal{H}$ is defined by*

$$B_Q(x) = sg\left[\mathop{\mathbf{E}}_{h\sim Q} h(x)\right]$$

where $sg(a) = 1$ if $a > 0$ and -1 otherwise

The risk of the Bayes classifier B_Q over a distribution D' on $(\mathcal{X} \times \mathcal{Y})$ is as follows.

Definition 2. *The risk of the Bayes classifier B_Q over Q on $\mathcal{H}$, is defined as the expected zero one loss over D', a distribution on $(\mathcal{X} \times \mathcal{Y})$.*

$$R_{D'}(B_Q) = \mathop{\mathbf{E}}_{(x,y)\sim D'} I\left(\mathop{\mathbf{E}}_{h\sim Q} y \cdot h(x) < 0\right).$$

where $I(a) = 1$ is the predicate a is true, and 0 is not.

We eventually outline some basic definitions in order to clarify the contributions of this work ahead in the paper : agreement ratio, margin and Kullback-Leibler divergence.

Definition 3. *The agreement ratio between two voters $h, h' \in \mathcal{H}$ is defined as*

$$\tau(h, h') = \frac{1}{m}\sum_{i=1}^{m} h(x_i)h'(x_i).$$

Definition 4. *The margin of a single voter $h \in \mathcal{H}$ is $\gamma(h) = \frac{1}{m}\sum_{i=1}^{m} y_i h(x_i)$.*

Definition 5. *The Kullback-Leibler (KL) divergence between distributions Q and P over $\mathcal{H}$ is defined by*

$$KL(Q||P) = \mathop{\mathbf{E}}_{h\sim Q} ln\frac{Q(h)}{P(h)}.$$

2.2 The $\mathcal{C}$-Bound & PAC-Bayesian guarantees

We state here the main context of our work by presenting the $\mathcal{C}$-bound and its properties, as first introduced in [10]. Let us first define one core concept : the margin of the majority vote.

Definition 6. *Given an example $x \in \mathcal{X}$ and its label $y \in \mathcal{Y}$ drawn according to a distribution D', $M_Q^{D'}$ is the random variable that gives the margin of the majority vote B_Q, defined as $M_Q(x,y) = \mathop{\mathbf{E}}_{h\sim Q} y \cdot h(x)$.*

The empirical margin is positive if B_Q classifies the example with the right class. From the margin, the $\mathcal{C}$-bound is presented in [10] as follows.

Definition 7. *For any distribution Q on a set $\mathcal{H}$ of functions valued in $\{-1,1\}$, for any distribution D' on $\mathcal{X} \times \mathcal{Y}$, let $\mathscr{C}_Q^{D'}$ be the $\mathcal{C}$-bound of B_Q over D', defined as*

$$\mathscr{C}_Q^{D'} = 1 - \frac{\left(\mu_1(M_Q^{D'})\right)^2}{\mu_2(M_Q^{D'})},$$

with $\mu_1(M_Q^{D'})$ being the first moment of the margin

$$\mu_1(M_Q^{D'}) = \mathop{\mathbf{E}}_{(x,y)\sim D'} M_Q(x,y),$$

and $\mu_2(M_Q^{D'})$ being the second moment of the margin

$$\mu_2(M_Q^{D'}) = \mathop{\mathbf{E}}_{(x,y)\sim D'} \left[M_Q(x,y)\right]^2 = \mathop{\mathbf{E}}_{(x,y)\sim D'} \left[\mathop{\mathbf{E}}_{h\sim Q} h(x)\right]^2$$

where the last equality comes from the fact that $y \in \{-1,1\}$, so $y^2 = 1$.

The following theorem, established and proven in [10], shows that the $\mathcal{C}$-bound is an upper-bound for the risk of the majority vote classifier.

Theorem 1. *For any distribution Q on a set $\mathcal{H}$ of functions valued in $\{-1,1\}$, and for any distribution D' on $\mathcal{X} \times \mathcal{Y}$, if $\mu_1(M_Q^{D'}) > 0$, we have $R_{D'}(B_Q) \leq \mathscr{C}_Q^{D'}$.*

Theorem 1 means that by minimizing the empirical $\mathcal{C}$-bound, an algorithm indirectly minimizes the risk of the Bayes majority vote B_Q, which is the main idea of MinCq [9] and CqBoost [19], presented in Section 2.3.

In terms of generalization guarantees, the PAC-Bayesian framework [16] provides a way to bound the true risk of B_Q, given a prior distribution P and a posterior distribution Q.

The following theorem, established in [19], gives an upper bound of the risk of the majority vote, which depends on the first and second moments of the margin as introduced in definitions 6 and 7.

Theorem 2. *For any distribution D on $\mathcal{X} \times \mathcal{Y}$, for any set $\mathcal{H}$ voters $h : \mathcal{X} \to [-1,1]$, for any prior distribution P on $\mathcal{H}$ and any $\delta \in]0,1]$, with a probability at least $1-\delta$ over the choice of the m-sample $\mathcal{S} \sim D^m$, and for every posterior distribution Q on $\mathcal{H}$, we have*

$$R_{D'}(B_Q) \leq 1 - \frac{\left(max(0, \underline{\mu_1})\right)^2}{min(1, \overline{\mu_2})}, \text{ where :}$$

$$\underline{\mu_1} = \mu_1(M_Q^S) - \sqrt{\frac{2}{m}\left[KL(P||Q) + ln\left(\frac{2\sqrt{m}}{\delta/2}\right)\right]}$$

$$\overline{\mu_2} = \mu_2(M_Q^S) + \sqrt{\frac{2}{m}\left[2KL(Q||P) + ln\left(\frac{2\sqrt{m}}{\delta/2}\right)\right]}.$$

Theorem 2 is an additional argument to advocate algorithms minimizing the $\mathcal{C}$-bound for it provides major generalization guarantees to these algorithms returning votes with a small empirical $\mathcal{C}$-bound value and a small divergence between prior and posterior distributions of the set $\mathcal{H}$.

2.3 MinCq & CqBoost

Let us focus on two algorithms that rely on the minimization of the $\mathcal{C}$-bound in order to learn an accurate ensemble of classifiers. MinCq [9] finds the weights that minimize the empirical $\mathcal{C}$-bound on a given set of voters, and CqBoost [19] inversely uses column generation and boosting to iteratively build a set of voters by minimizing the $\mathcal{C}$-bound.

2.3.1 MinCq

MinCq's principle is creating a majority vote from a finite set of voters. In order to find the optimal weights on these voters, MinCq aims at finding the ones that minimize the $\mathcal{C}$-bound. To avoid overfitting, MinCq considers a restriction on the posterior distribution the authors called quasi-uniformity [9], and adds an equality constraint on the first moment of the margin.

Given a set of voters $\mathcal{H}$, MinCq actually finds the distribution on $\mathcal{H}$ that minimizes the $\mathcal{C}$-bound. Beneficially, it is then ensured to perform as well in train as in generalization, according to Theorems 1 and 2. The main drawback of MinCq is its computational training time : its algorithmic complexity is $\mathcal{O}(m \times n^2 + n^3)$, where n is the number of voters, which prevents it from scaling up to large datasets.

2.3.2 CqBoost

Like MinCq, CqBoost is an algorithm based on the minimization of the $\mathcal{C}$-bound. It was designed to fasten the minimization performed by MinCq. It is actually a sparser $\mathcal{C}$-bound minimization algorithm, hence enabling better interpretability and faster training and prediction processes.

CqBoost is based on a column generation process that iteratively builds the vote, voter by voter. It is similar to boosting in the way that, at each iteration t, the choice of the voter to add is made by optimizing the edge criterion [4]. Once a voter has been added to the current selected voters set H_t, CqBoost finds the optimal weights over H_t by minimizing the $\mathcal{C}$-bound similarly to MinCq, by solving a quadratic program.

The sparsity of CqBoost is explained by its ability to stop the iterative vote building early enough to avoid overfitting. Nevertheless, though CqBoost is faster and sparser than MinCq, it is still not applicable to big data for its algorithmic complexity is still $\mathcal{O}(m \times T^2 + T^3)$ where T is the number of boosting iterations.

Next sections present an algorithm faster than both MinCq and CqBoost, which also preserves the sparsity of CqBoost as well as the expected accuracy.

3 Fast $\mathcal{C}$-Bound minimization algorithm

We present here the two-voters $\mathcal{C}$-bound minimization problem and its solution, which are central in CB-Boost. Then, we introduce CB-Boost's pseudo-code in Algorithm 1, illustrating how it is a fast $\mathcal{C}$-bound minimization algorithm.

3.1 Two-voters minimization problem

One core of the CB-Boost algorithm is a specific case of $\mathcal{C}$-bound minimization, where the set of voters $\mathcal{H}$ is made up of only two voters. Let $\mathcal{H} = \{f, g\}$, and $\pi \in]0,1[$: the empirical $\mathcal{C}$-bound with a distribution $Q = \{\pi, 1-\pi\}$ over $\mathcal{H}$ and a uniform distribution D' over S is

$$1 - \frac{1}{m} \frac{\left(\sum_{i=1}^{m} \left[y_i \left(\pi f(x_i) + (1-\pi) g(x_i) \right) \right] \right)^2}{\sum_{i=1}^{m} \left(y_i \left[\pi f(x_i) + (1-\pi) g(x_i) \right] \right)^2} . \quad (1)$$

This expression is referred to as the two-voters $\mathcal{C}$-bound. The two-voters specific empirical $\mathcal{C}$-bound minimization problem is then to compute $\pi \in]0,1[$ that minimizes Equation 1.

In order to simplify analysis to go, the variable $\pi \in]0,1[$ of Equation 1 is replaced by $\beta = \frac{1}{\pi} - 1$ which ranges in $\mathbb{R}_+^*$. Then, as proven in supplementary material, a new problem equivalent to the two-voters empirical $\mathcal{C}$-bound minimization is stated in definition 8.

Definition 8. *For f, g two voters and for all $\beta \in \mathbb{R}_+^*$, the two-voters empirical $\mathcal{C}$-bound minimization is solved with β^* such that :*

$$\begin{aligned} \beta^* &= \arg\min_{\beta \in \mathbb{R}_+^*} \Phi_{f,g}(\beta) \\ &= \arg\min_{\beta \in \mathbb{R}_+^*} 1 - \frac{1}{m} \frac{\left(\sum_{i=1}^{m} \left[y_i \left(f(x_i) + \beta g(x_i) \right) \right] \right)^2}{\sum_{i=1}^{m} \left(y_i \left[f(x_i) + \beta g(x_i) \right] \right)^2} \end{aligned}$$

A technical development of the optimization problem leads to an analytical solution made up of two possible values for β^*, as stated by Theorem 3.

Theorem 3.

$$\beta^* = \arg\min_{\beta \in \mathbb{R}_+^*} \Phi_{f,g}(\beta) = \begin{cases} \frac{-C_1 \pm \sqrt{C_1^2 + 4C_0C_2}}{2C_2} \\ \frac{-C_1}{2C_2} \end{cases}$$

$$C_0 = A_1B_0 \ ; \ C_1 = 2(A_0B_2 - A_2B_0) \ ; \ C_2 = A_1B_2 - A_2B_1 \ ;$$
$$B_2 = \|f\|_2^2 \ ; \ B_1 = 2m\tau(f,g) \ ; \ B_0 = m$$
$$A_2 = m^2\gamma(g)^2 \ ; \ A_1 = 2m^2\gamma(g)\gamma(f) \ ; \ A_0 = m^2\gamma(f)^2$$

Proof. The full proof is provided in supplementary material : this minimization problem is equivalent to finding the roots of a quadratic equation in β. □

3.2 Main algorithm : step-by-step $\mathcal{C}$-Bound minimization

Let us present the overall sketch of the CB-Boost algorithm which is intended to minimize the risk of the majority votes through the iterative simpler minimization of the two-voters $\mathcal{C}$-bound presented in Theorem 3. As CqBoost (Section 2.3), the CB-Boost algorithm relies on a column generation technique to span a set of hypotheses. To do so, we build an algorithm over the two voters problem that is a greedy minimization of the $\mathcal{C}$-bound. The principle is to optimize the weights of the majority vote one voter at a time. In order to initialize CB-Boost, we use $g_0 \in \mathcal{H}$ the hypothesis with the best margin $\gamma(g_0)$. This aims at fastening the convergence of CB-Boost by starting the vote building with the strongest available voter in terms of margin.

At each iteration t, the optimal weight β_t is obtained by selecting the voter g_t that gives the lowest minimum

Algorithm 1 CB-Boost

Require:

1: $g_0 \leftarrow$ max. margin # Find g_0 the voter of $\mathcal{H}$ with the highest margin $\gamma(g_0)$
2: $F_0 \leftarrow g_0$ # Initialize the majority vote with g_0
3: **for** $t \leftarrow 0, \ldots, T-1$ **do**
4: $\quad g_{t+1}, \beta_{t+1} \leftarrow \underset{j \in [\![1,n]\!], \beta_j \in \mathbb{R}_+^*}{\arg\min} \left(1 - \frac{1}{m} \frac{\left(\sum_{i=1}^m [y_i(F_t(x_i)+\beta_j f_j(x_i))]\right)^2}{\sum_{i=1}^m [y_i(F_t(x_i)+\beta_j f_j(x_i))]^2}\right)$
5: $\quad F_{t+1} \leftarrow F_t + \beta_{t+1} g_{t+1}$
6: **end for**
7: **return** $\hat{y} = \mathrm{sg}\left(\sum_{j=0}^T \beta_j g_j(x)\right)$

of the two-voters $\mathcal{C}$-bound. The problem solved at Line 4 of Algorithm 1 is the two-voters problem (definition 8) applied to the previous iteration's majority vote and the potential new voter. Hence, it is worthwhile to notice that Lines 4 and 5 of the algorithm are closely intertwined : (1) the majority vote F_{t+1} is computed at Line 5 from the previous majority vote F_t and the weight β_{t+1} and voter g_{t+1} selected Line 4, then (2) this majority vote F_{t+1} is used during the next iteration, as one of the voter to be considered in the empirical two-voters $\mathcal{C}$-bound minimization (Line 4, next step).

This way of building a majority vote is equivalent to optimizing the $\mathcal{C}$-bound on $\mathcal{H}$ with a gradient descent algorithm that greedily optimizes the majority vote one weight at a time. Consequently, at the beginning, the weights on $\mathcal{H}$ will be a Dirac distribution with the best-margin hypothesis's weight being the only one non-zero, and at each iteration $t+1$ one more element of $\mathcal{H}$ will have a non-zero weight β_{t+1}, computed in order to minimize the $\mathcal{C}$-bound on $\mathcal{H}$.

One great appeal of CB-Boost is the simplicity of the two voters $\mathcal{C}$-bound minimization. Indeed, the algorithmic complexity of CB-Boost depends on the number of iterations T, of examples m, and of generated columns n. The two-voters problem is solved in $\mathcal{O}(n \times m)$. So CB-Boost's complexity is $\mathcal{O}(n \times m \times T)$.

3.3 Remarks

3.3.1 On the voter choice criterion

In most boosting algorithms, the criterion used to choose the next voter to add to the majority vote is the edge over the distribution on the examples [4]. The edge is the dual of the margin ; it measures the ability for the considered voter to classify the hardest examples.

However, as the objective quantity in our algorithm is the $\mathcal{C}$-bound, and because our minimization problem can be solved quickly for all available voters, we do not use these to find the best voter. Instead, we compute the minimum $\mathcal{C}$-bound for each possible voter and select the one that gives the lowest minimum.

3.3.2 On the $\mathcal{C}$-bound indirect example reweighting

In order to bring diversity in the majority vote, Adaboost [7] updates weights over the examples at each iteration, to exponentially emphasize the examples on which the algorithm fails.

In CB-Boost, the $\mathcal{C}$-bound, by considering both the first and second moments of the margins, take in account the individual performance (first moment) of each voter and their disagreement (second moment). Therefore, minimizing the $\mathcal{C}$-bound requires to keep a trade-off between maximizing the margin of the vote and its internal disagreement. This is the reason why CB-Boost does no include any example weighting. The effect of each component of the $\mathcal{C}$-bound in the framework of CB-Boost is given in supplementary materials.

3.3.3 On the difference between the majority votes

Intuitively, the concession made in CB-Boost to fasten CqBoost is on the weights of the majority vote. Indeed, CB-Boost's output voters weights are not the ones that truly minimize the $\mathcal{C}$-bound. In contrast, CqBoost returns the majority vote that minimizes the $\mathcal{C}$-bound, for the considered set of voters.

Hence, with CB-Boost, the returned voters weights are sub-optimal because they have been optimized greedily throughout the iterations. Nevertheless, the $\mathcal{C}$-bound computed during the training phase is the true one for the considered majority vote, which explains the theoretical results of next section.

4 Theoretical results

4.1 The $\mathcal{C}$-Bound decreases : quantification

Let us analyze the behavior of the $\mathcal{C}$-bound of the majority vote build by CB-Boost during its optimization by CB-Boost : in the end, the main result is that the empirical $\mathcal{C}$-bound decreases with iterations.

We first analyze our variables and stipulate four conditions on them that are experimentally non-restrictive. In the following, we will note F the vote built by CB-Boost, and g a potential new voter to add at the next iteration.

Property 1. *We know that*

- $1 > \gamma(g) \geq \frac{1}{m}$, *because g is supposed to be a weak classifier that has an empirical risk* $< \frac{1}{2}$,
- $\gamma(g) < \gamma(F)$, *as the majority vote F is assumed to be better than the weak classifier g.*

Proof. Provided in supplementary materials. □

We then state crucial experimental conditions and assumptions which underlie the proofs of the final results. They all hold in all experiments we drove, but the proofs are currently unachieved. With $\|F\|_2^2 = \sum_{i=1}^{m} F(x_i)^2$, we conjecture that :

$$m\gamma(F)^2 > \|F\|_2^2\,\gamma(g)^2 \quad (2) \qquad \|F\|_2^2 > m \quad (4)$$

$$\gamma(F) > 2\gamma(g)\tau(F,g) \quad (3) \qquad -1 < \tau(F,g) < 1 \quad (5)$$

With such definitions and equations, we are now able to characterize the optimal weights that solve the two-voters $\mathcal{C}$-bound minimization problem.

4.2 The computational advantage

The solution of the $\mathcal{C}$-bound minimization in our context is the first solution stated by Theorem 3 :

Property 2. *The weight that minimizes the $\mathcal{C}$-bound is* $\beta = \frac{-C_1+\sqrt{C_1^2+4C_0C_2}}{2C_2}$.

Proof. The full proof is provided in supplementary materials, but to sum it up, as we have seen in Theorem 3, β is either $\frac{-C_1\pm\sqrt{C_1^2+4C_0C_2}}{2C_2}$ or $\frac{-C_1}{2C_2}$.

- If $\beta = \frac{-C_1}{2C_2}$, assumptions 4 and 5 are violated. Hence, $\beta \neq \frac{-C_1}{2C_2}$.
- If $\beta = \frac{-C_1-\sqrt{C_1^2+4C_0C_2}}{2C_2}$, assumption 2 is violated. □

So we are able to get the exact form of the solution of the two-voters $\mathcal{C}$-bound minimization problem, which depends on the majority vote built in previous iterations and the considered voter. Here is the main computational advantage of CB-Boost over CqBoost and MinCq because these latters both have to use a quadratic problem solver to find the majority vote that minimizes the $\mathcal{C}$-bound.

It is worthwhile noticing that the $\mathcal{C}$-bound computed here is the exact value of the $\mathcal{C}$-bound on the distribution obtained by normalizing the weights $\{\beta_j\}_{j=1}^{T}$ on the voters returned by CB-Boost.

4.3 The empirical $\mathcal{C}$-Bound decreases

In this part, we show that at each iteration of CB-Boost, the $\mathcal{C}$-bound of the majority vote built by the algorithm decreases :.

Property 3. *If $F_t = \sum_{j=0}^{t} \beta_j g_j$ is the majority vote built at iteration t by CB-Boost, and $\mathscr{C}_t^F$ is the empirical $\mathcal{C}$-bound of F_t defined by $\mathscr{C}_t^F = \left(1 - \frac{\left(\sum_{i=1}^{m}[y_iF_i]\right)^2}{\sum_{i=1}^{m}(y_iF_i)^2}\frac{1}{m}\right)$, and similarly F_{t+1}, and $\mathscr{C}_{t+1}^F$ for iteration $t+1$, then $\mathscr{C}_{t+1}^F \leq \mathscr{C}_t^F$*

Proof. Let $\mathscr{C}_t^F$, $\mathscr{C}_{t+1}^F$ be the $\mathcal{C}$-bounds of F_t and F_{t+1} respectively. Yet, $F_{t+1} = F_t + \beta_{t+1}g_{t+1}$ with β_{t+1} being, by definition, the weight that minimizes $\Phi_{F_t,g_{t+1}}$. We then have

$$\begin{aligned}\mathscr{C}_{t+1}^F &= \Phi_{F_t,g_{t+1}}(\beta)\\ &\leq \left(1 - \frac{\left(\sum_{i=1}^{m}[y_iF_i]\right)^2}{\sum_{i=1}^{m}(y_iF_i)^2}\frac{1}{m}\right) = \Phi_{F_t,g_{t+1}}(0) = \mathscr{C}_t^F\,.\end{aligned}$$

□

Let us now quantify the decrease rate of the empirical $\mathcal{C}$-bound for each iteration of CB-Boost, depending on the previous majority vote and the considered voter.

Property 4. *Between iterations t and $t+1$ of CB-Boost, the $\mathcal{C}$-bound decreases exactly by* $\frac{\left(\gamma(g)\|F\|_2^2-\gamma(F)m\tau(F,g)\right)^2}{\|F\|_2^2\left(\|F\|_2^2-m\tau(F,g)^2\right)}$.

Proof. The full proof is provided in supplementary material. A sketch it is : (1) Get the value of the arg minimum β, (2) Compute the difference between $\mathscr{C}_t^F$ and $\mathscr{C}_{t+1}^F$ as a function of β, and (3) Replace with the value of β, and simplify. □

	CqBoost	CB-Boost	Adaboost	MinCq	SCM
australian	**0.144 ± 0.014**	**0.143 ± 0.009**	**0.145 ± 0.011**	**0.141 ± 0.008***	**0.146 ± 0.009**
bupa	**0.332 ± 0.032**	**0.335 ± 0.033**	**0.330 ± 0.029***	**0.350 ± 0.039**	0.413 ± 0.036
cylinder	**0.266 ± 0.027**	**0.270 ± 0.021**	**0.251 ± 0.016***	**0.261 ± 0.019**	0.328 ± 0.020
hepatitis	**0.390 ± 0.053**	**0.382 ± 0.019**	0.404 ± 0.039	**0.342 ± 0.043***	0.404 ± 0.059
ionosphere	0.140 ± 0.021	0.160 ± 0.014	**0.101 ± 0.017***	0.132 ± 0.019	0.106 ± 0.038
yeast	**0.294 ± 0.018**	**0.299 ± 0.009**	**0.295 ± 0.015**	**0.289 ± 0.017***	0.303 ± 0.005
balance	0.053 ± 0.020	0.055 ± 0.016	**0.015 ± 0.007***	0.053 ± 0.014	0.301 ± 0.023
pima	**0.242 ± 0.015**	**0.235 ± 0.018***	**0.248 ± 0.020**	0.250 ± 0.016	0.277 ± 0.016
MNist-0v9	0.015 ± 0.002	**0.012 ± 0.001***	**0.013 ± 0.001**	**0.012 ± 0.002**	0.072 ± 0.031
MNist-5v3	**0.080 ± 0.009**	**0.083 ± 0.008**	**0.075 ± 0.008***	**0.076 ± 0.008**	0.192 ± 0.048
MNist-5v6	0.040 ± 0.002	0.041 ± 0.003	0.043 ± 0.004	**0.036 ± 0.003***	0.123 ± 0.017
MNist-7v9	**0.072 ± 0.005**	**0.073 ± 0.004**	**0.072 ± 0.005**	**0.071 ± 0.002***	0.095 ± 0.011
Awa-TvW	0.093 ± 0.003	**0.081 ± 0.001***	**0.085 ± 0.013**	**0.09 ± 0.011**	0.225 ± 0.013

TABLE 1 – Zero one loss on test, a bold result means that, considering the standard deviation, the algorithm performed as well as the best mean. A starred result means that the classifier returned the best mean result.

4.4 Getting the training error bound and generalization guarantees

The training error is bounded We derive here a training error bound from the work of [10]. Theorem 1 establishes that the $\mathcal{C}$-bound is an upper-bound of the training error for the built majority vote. We then can derive a lemma from Theorem 1 and Property 4.

Lemma 1. *The risk of the majority vote, built by CB-Boost at iteration t is bounded by*

$$\mathscr{C}_1^F - \sum_{j=2}^{t} \mathscr{S}_j$$

with $\mathscr{S}_j$ being the quantity introduced in Property 4

$$\mathscr{S}_j = \frac{\left(\gamma(g_j)\,\|F_j\|_2^2 - \gamma(F_j) m\tau(F_j, g_j)\right)^2}{\|F_j\|_2^2 \left(\|F_j\|_2^2 - m\tau(F_j, g_j)^2\right)}\,.$$

Proof. The proof is straightforward by combining Theorem 1 and Property 4. □

This training error bound allows to assess CB-Boost's capacity to learn relevant models based on the available voters.

4.4.1 Generalization guarantees

Section 2.2 presents a PAC-bound that gives generalization guarantees when minimizing the $\mathcal{C}$-bound. These guarantees are tighter when the $\mathcal{C}$-bound of a majority vote is low, which is exactly what CB-Boost aims at returning. However, as seen in 2.2, returning a majority vote with a small $KL(Q||P)$ is essential in order to have good generalization guarantees. In [19], the authors established that if the number of prior voters is far lower than the number of examples, $n << m$, then minimizing $KL(Q||P)$ is negligible in comparison with minimizing the $\mathcal{C}$-bound of the majority vote.

If the case $n << m$ is not applicable, we need to characterize intuitively $KL(Q||P)$. Indeed, $KL(Q||P)$ represents the complexity of the built model using Q. So as the iterations go, the $\mathcal{C}$-bound decreases whereas $KL(Q||P)$ increases. Then, in order to keep $KL(Q||P)$ low enough for the bound of Theorem 2, it is relevant to use early-stopping by choosing a maximal number of iterations to perform. Consequently, in order to keep the generalization guarantees, we set the maximum number of iterations on CB-Boost as an hyper-parameter that can be chosen using hold-out data.

5 Experimentation

In this section we empirically analyze CB-Boost's quality in terms of accuracy, sparsity and computational learning time, on multiple datasets, compared to several state-of-the art algorithms.

5.1 Performance in zero-one loss

Protocol : *Datasets* We used the following datasets from the UCI repository [5] to test CB-Boost's capacity to solve simple problems with few examples and/or a low dimensionality. We chose Australian (690 × 14), Balance (625 × 4), Bupa (345 × 6), Cylinder (540 × 35), Hepatitis (155 × 19), Ionosphere (351 × 34), Pima (768 × 8), Yeast (1484 × 8). On every dataset, we generated 10 pairs of complementary decision stumps for each attribute as base voters and used one half of the dataset to train and the other half to test.

To be able to analyze CB-Boost's behavior on larger data, we used the M-Nist dataset [14]. We selected four difficult couple of classes : (0,9), (5,3), (5,6) and (7,9). As MNist has 784 attributes, we used 1 pair of

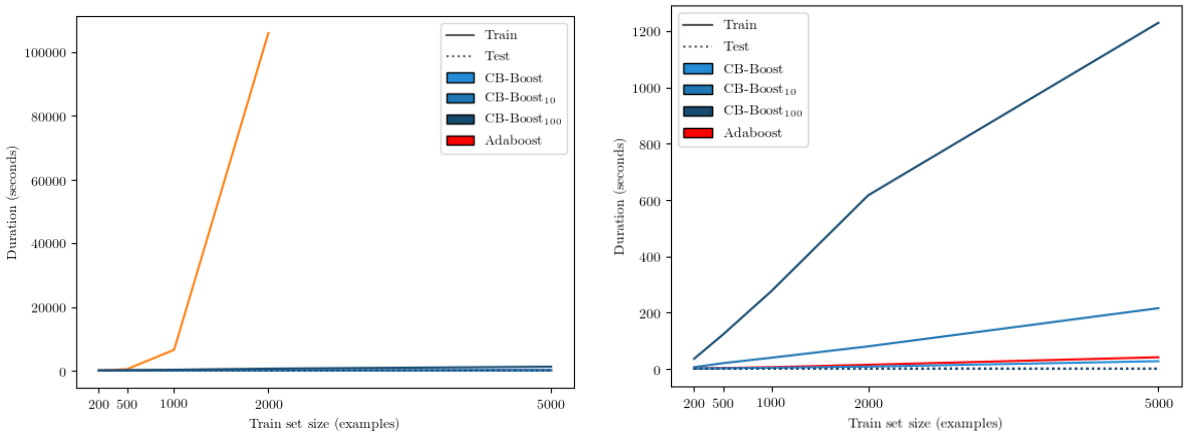


FIGURE 1 – Computational efficiency with Cq-Boost (left), and without it (right).

complementary decision stumps for each attribute as base voters. We used 4% of the dataset ($\sim$ 471 ex.) to train a the remaining examples ($\sim$ 11000 ex.) to test.

Finally, to have a dataset with a large number of attributes, we used the Animals with Attributes dataset [11] by fusing two descriptors (surf-hist and cq-hist) generating a dimentionality of 4688 attributes. As each attribute on its own is barely relevant, we generated 2000 decision trees of depth 3 as base voters. These trees where learned on randomly sub-sampled attributes (60% of the original dimension, with replacement). We selected two classes : tiger and wolf, that are very close are will provide some challenge to differentiate. We denote this dataset AwA-TvW. On AwA, we used one half of the dataset to train (546 ex.) and the other half to test (546ex.)

Classifiers The benchmark that we used is composed of CqBoost, Adaboost [7], SCM [15], MinCq, CB-Boost. For each classifier, hyper-parameters where found with a randomized search with 30 draws (for each hyper-parameter), over a distribution that incorporates prior knowledge on the algorithm, but is independent on the dataset. They were validated by a 5-folds cross-validation and each experiment was realized ten times, for a better stability. Table 1 reports the mean and standard deviation over these.

Results For all datasets except Ionosphere and Balance, CB-Boost is performing at least as well as the state-of-the art, considering the standard deviation. On both the other, Adaboost performs better than all the algorithms based on the $\mathcal{C}$-bound. Indeed, CqBoost's, MinCq's and CB-Boost's results are fairly equivalent.

On the four MNist datasets, results are similar, except for the SCM, whose sparsity undermines the performance on complex data. Indeed, the differentiation between all the other algorithms is made on less than 1% of the testing set. Nevertheless, we can note that on 3 datasets out of 4, CB-Boost is among the best classifiers and on Mnist-0v9, it returns the best mean. On AwA-TvW, CB-Boost also has the best mean of all the studied algorithms.

To conclude this experiment, CB-Boost is competitive with the state-of-the-art in terms of zero-one loss 9 times out of 12, and has the best mean 3 times.

5.2 Computational efficiency

Protocol In Figure 1, we analyze the computational efficiency of CB-Boost on the M-Nist-0v9 dataset, for one whole training with 400 iterations for Adaboost, CB-Boost, and CqBoost. All the algorithms were run on one thread with increasingly large training sets, to be able to judge their pure efficiency.

Results Figure 1 shows that CqBoost does not scale up with a large number of training examples, we did not use more than 2000 examples, as it was not relevant.

In the left sub-figure, we see that CB-Boost is slightly faster than Adaboost when provided with the same number of stumps (one per attribute). We also plotted the duration of a training phase with CB-Boost when provided with 10 and 100 stumps per attribute (respectively labelled CB-Boost$_{10}$ and CB-Boost$_{100}$) and we can see that CB-Boost's theoretical complexity holds in the experiments.

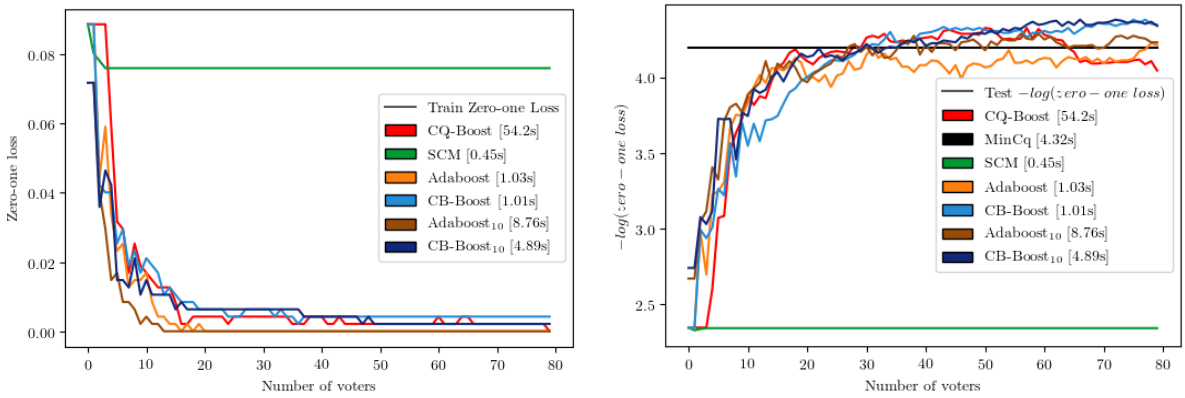


FIGURE 2 – Zero-one loss on train (left) and $-log(zero\text{-}one\ loss)$ on test (right) throughout the learning process.

5.3 Sparsity

Protocol In this experiment, we analyze the convergence speed of several ensemble methods with optimal hyper-parameters which concerns the final number of voters in the set. The chosen dataset is MNIST 0v9 as it is more complex than UCI datasets but still small enough for CqBoost and MinCQ to have reasonable computing time on it. We use 4% of the dataset (471 ex.) for training, and the remaining 96% (11320 ex.) for testing. The analyzed algorithms are, besides CB-Boost : MinCq, CqBoost, SCM[15] and Adaboost [7].

As CB-Boost and Adaboost are fast-learning algorithms, we were able to run two different settings of column generations on MNist : 1 stump per attribute, and 10 per attribute, whereas for CqBoost and MinCq only the first setting was tested, considering the computational time, even on multiple threads.

Results In Figure 2, we see that, even if CB-Boost converges a bit less quickly than Adaboost and CqBoost on the training set, its performance on the test set is slightly better than all the other algorithms. So, on the point of view of the tradeoff between sparsity and performance, CB-Boost is better that the others.

5.4 Comparison to MinCq

We compare here CB-Boost to MinCq in order to evaluate empirically the loss of performance caused by the greedy approach to minimize the $\mathcal{C}$-bound.

Protocol We used MNist 0v9 with 4% of it (471 ex.) to train and the remaining (11320 ex.) to test. We used predefined hyper-parameters : for CB-Boost, the maximum number of iterations was 200, and for MinCq, the margin parameter was 0.05. To analyze the difference between the majority votes returned by MinCq and CB-Boost, at each iteration of CB-Boost, we computed the optimal majority vote in term of $\mathcal{C}$-bound with MinCq on the set of voters chosen by CB-Boost and evaluated its performance. To assess the overall capacity of CB-Boost to return a relevant majority vote, we used MinCq to compute the best majority vote, considering all possible voters.

Results In Figure 3, we see that there is a slight, but noticeable difference between the $\mathcal{C}$-bound of CB-Boost's majority vote and MinCq's one. Moreover, in terms of training performance, the difference is even smaller, and for test, CB-Boost's is even as good as MinCq's, implying the possibility that MinCq overfits the empirical value of the $\mathcal{C}$-bound. Concerning the optimal vote in terms of $\mathcal{C}$-bound, considering the 784×2 self complemented base learners, represented by the green line in Figure 3, we can see that in training, MinCq and CB-Boost converge toward it at the same rate, event if MinCq reaches it faster it tends to move away from it as CB-Boost closes the gap.

6 Conclusion & Future work

In this paper, we presented a greedy $\mathcal{C}$-bound minimization based on CqBoost [19] that, while maintaining its sparsity and accuracy properties, has a much lighter computational demands. This algorithm keeps the training and generalization guarantees given by the $\mathcal{C}$-bound [10] and has the interesting property to allow a quantification of the decrease of its bound. Experimentally, it is competitive with the state of the art

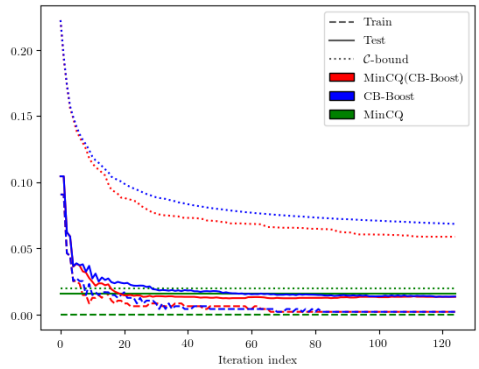


FIGURE 3 – Zero-one-loss and $\mathcal{C}$-bound for MinCq and CB-Boost. The red curves show the behavior of MinCq's majority vote on the set of voters proposed by CB-Boost during training. The green one shows MinCq's performance on all the attributes.

in terms of performance, computational efficiency and sparsity.

In future work, we will analyze deeper theoretical properties of CB-Boost, focusing on its dual form, and on finding a stopping criterion.

Références

[1] L. Breiman. Bagging predictors. Mach. Lear., 24(2) :123–140, Aug 1996.

[2] L. Breiman. Random forests. Mach. Lear., 45(1) :5–32, Oct 2001.

[3] O. Catoni. Pac-bayesian supervised classification : the thermodynamics of statistical learning. arXiv :0712.0248, 2007.

[4] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. Mach. Lear., 46(1) :225–254, Jan 2002.

[5] D. Dua and C. Graff. UCI ml repo., 2017.

[6] G. K. Dziugaite and D. M. Roy. Data-dependent pac-bayes priors via differential privacy. In Adv. in NIPS, pages 8440–8450, 2018.

[7] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Jour. of Comp. and Sys. Sci., 55(1) :119–139, 1997.

[8] P. Germain, A. Lacasse, F. Laviolette, and M. Marchand. Pac-bayesian learning of linear classifiers. In Proc. 26th ICML, pages 353–360. ACM, 2009.

[9] P. Germain, A. Lacasse, F. Laviolette, M. Marchand, and J.-F. Roy. Risk bounds for the majority vote : From a pac-bayesian analysis to a learning algorithm. JMLR, 16(1) :787–860, Jan. 2015.

[10] A. Lacasse, F. Laviolette, M. Marchand, P. Germain, and N. Usunier. Pac-bayes bounds for the risk of the majority vote and the variance of the gibbs classifier. pages 769–776, 01 2006.

[11] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In IEEE CVPR, pages 951–958. IEEE, 2009.

[12] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. JMLR, 5 :27–72, Dec. 2004.

[13] J. Langford and J. Shawe-Taylor. Pac-bayes & margins. In Adv. in NIPS, :439–446, 2003.

[14] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[15] M. Marchand and J. S. Taylor. The set covering machine. JMLR, 3 :723–746, Mar. 2003.

[16] D. A. McAllester. Some pac-bayesian theorems. Mach. Lear., 37(3) :355–363, Dec 1999.

[17] D. A. McAllester. Pac-bayesian stochastic model selection. Mach. Lear., 51(1) :5–21, Apr 2003.

[18] E. Parrado-Hernández, A. Ambroladze, J. Shawe-Taylor, and S. Sun. Pac-bayes bounds with data dependent priors. JMLR, 13 :3507–3531, 12 2012.

[19] J.-F. Roy, M. Marchand, and F. Laviolette. A column generation bound minimization approach with pac-bayesian generalization guarantees. In A. Gretton and C. C. Robert, editors, Proc. 19th AIStats, volume 51 of Proc. of MLR, pages 1241–1249, Cadiz, Spain, 09–11 May 2016. PMLR.

[20] M. Seeger. Pac-bayesian generalisation error bounds for gaussian process classification. JMLR, 3(Oct) :233–269, 2002.

[21] Y. Seldin, N. Cesa-Bianchi, P. Auer, F. Laviolette, and J. Shawe-Taylor. Pac-bayes-bernstein inequality for martingales and its application to multiarmed bandits. In Proc. of the Work. On-line Trad. of Expl. & Exploit. 2, pages 98–111, 2012.

[22] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. JMLR, 7 :1531–1565, 2006.

[23] L. G. Valiant. A theory of the learnable. Commun. ACM, 27(11) :1134–1142, 1984.

A Supplementary Material : Algorithms

Here, we will present shortly MinCq [9] and CqBoost [19]. As for the notations, we used :

— $\mathbf{F} \triangleq \begin{bmatrix} f_1(x_1) & f_2(x_1) & \dots & f_{2n}(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_{2n}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_m) & f_2(x_m) & \dots & f_{2n}(x_m) \end{bmatrix}$, as the vote matrix, containing the vote of each weak classifier in $\mathcal{H}$ and it's complementary $f_{n+j} = -f_j,\ j = 1..n$ on each example $x_i,\ i = 1..m$ of the training set $\mathcal{S}$,

— $\mathbf{q}$ as the weight vector on the voters,

— $\mathbf{1}_m$ as the unitary vector of size m

— μ as the minimum margin for the weak classifiers

$$\begin{aligned} \underset{\mathbf{q},\gamma}{\text{argmin}} \quad & \frac{1}{m}\gamma^\top\gamma \\ \text{subject to} \quad & \gamma = \text{diag}(\mathbf{y})\mathbf{F}\mathbf{q}, \quad \frac{1}{m}\mathbf{1}_m^\top\gamma \geq \mu, \\ & \mathbf{q} \geq \mathbf{0}_{2n}, \quad \mathbf{1}_{2n}^\top\mathbf{q} = 1. \end{aligned} \tag{6}$$

$$\begin{aligned} \underset{\boldsymbol{\alpha},\beta,v}{\text{argmin}} \quad & \frac{m}{4}\boldsymbol{\alpha}^\top\boldsymbol{\alpha} - \frac{\beta}{2}\mathbf{1}_m^\top\boldsymbol{\alpha} + \frac{\beta^2}{4} - \beta\mu + v \\ \text{subject to} \quad & \mathbf{F}^\top \text{diag}(\mathbf{y})\boldsymbol{\alpha} \leq v\mathbf{1}_{2n}, \\ & \beta \geq 0. \end{aligned} \tag{7}$$

B Supplementary material : On the $\mathcal{C}$-bound indirect example re-weighting, example

In this section, we will dissect the fact that minimizing the $\mathcal{C}$-bound contains an built-in example weighting through disagreement in the second moment of the margin.

We will analyze intuitively on an iteration of CB-Boost how the $\mathcal{C}$-bound orients the update of the majority vote.

As seen in Algorithm 1, the problem solved at each iteration is :

$$\underset{j\in[\![1,n]\!],\beta_j\in\mathbb{R}_+^*}{\arg\min} \left(1 - \frac{1}{m}\frac{\left(\sum_{i=1}^m [y_i(F_t(x_i) + \beta_j f_j(x_i))]\right)^2}{\sum_{i=1}^m [y_i(F_t(x_i) + \beta_j f_j(x_i))]^2}\right)$$

So let's suppose, we found β_{t+1}, and g that solve this problem. It is clear that minimizing the $\mathcal{C}$-bound is equivalent to finding the voter and weight that will give the best trade-off between maximizing the numerator

$$\left(\sum_{i=1}^m [y_i(F_t(x_i) + \beta_{t+1}g(x_i))]\right)^2$$

and minimizing the denominator

$$\sum_{i=1}^m [y_i(F_t(x_i) + \beta_{t+1}g(x_i))]^2$$

We will now analyze the effect of these two quantities on the majority vote :

— The numerator can be seen as $(\gamma(F_t)+\beta_{t+1}\gamma(g))^2$. it's function is intuitively to ensure that the chosen voter will have the largest possible margin.

— The denominator is

$$\begin{aligned} & \sum_{i=1}^m [y_i(F_t(x_i) + \beta_{t+1}g(x_i))]^2 \\ =& \sum_{i=1}^m F_t(x_i)^2 + 2\beta_{t+1}F_t(x_i)g(x_i) + {\beta_{t+1}}^2 g(x_i)^2 \\ =& \sum_{i=1}^m F_t(x_i)^2 + 2\beta_{t+1}\sum_{i=1}^m F_t(x_i)g(x_i) + {\beta_{t+1}}^2\sum_{i=1}^m 1^2 \\ =& \|F_t\|_2^2 + {\beta_{t+1}}^2 + 2\beta_{t+1}m\tau(F_t, g) \end{aligned}$$

So it has two functions, the first is a regularization one, with the two first terms of the sum, $\|F_t\|_2^2 + {\beta_{t+1}}^2$. And the second is to make sure that the added voter does not agree with the previous vote. The disagreement is a way to introduce variety in the vote and serves the same purpose as the re-weighting of examples in boosting. However, comparatively, in Adaboost, the examples that are heavily weighted are the one on which the classifier fails, whereas in CB-Boost the disagreement has a more general purpose of maintaining a diverse vote.

So, intuitively minimizing the $\mathcal{C}$-bound, CB-Boost keeps a tradeoff between a relevant and diverse set of voters. A possible future work would be to generate a distribution on the examples that will allow to fasten the convergence, without loosing the intuitive generality of disagreement.

C Supplementary Material : Proofs

Proof. **Proof of the equivalence of Definition 8** Just by changing the variable from $\pi \in]0, 1[$ to $\beta =$

Algorithm 2 MinCq

1: **Solve**

$$\begin{aligned} \underset{\mathbf{q}}{\text{argmin}} \quad & \frac{1}{m}\mathbf{q}^\top \mathbf{F}^\top \mathbf{F}\mathbf{q} \\ \text{subject to} \quad & \frac{1}{m}\mathbf{y}^\top \mathbf{F}\mathbf{q} = \mu, \quad \frac{1}{m}\mathbf{1}_m^\top \gamma \geq \mu, \\ & [\mathbf{I}_n \mathbf{I}_n]\,\mathbf{q} = \mu, \quad \mathbf{q} \geq \mathbf{0}_{2n}. \end{aligned}$$

2: **return** $\mathbf{q}$

Algorithm 3 CqBoost

Require: T, number of iterations

1: $\mathbf{q} \leftarrow \mathbf{0}_n$
2: $\boldsymbol{\alpha} \leftarrow \frac{1}{m}\mathbf{1}_m$
3: $\tilde{\mathbf{F}} \leftarrow$ empty matrix
4: **for** $t \leftarrow 1, \ldots, T$ **do**
5: $\quad i \leftarrow \text{argmax}_i \sum_{k=1}^m \alpha_k y_k F_{ki}$
6: $\quad$ **if** $\sum_{k=1}^m \alpha_k y_k F_{ki} \leq v + \epsilon$ **then**
7: $\quad\quad$ Leave loop
8: $\quad$ **end if**
9: $\quad$ Update $\tilde{\mathbf{F}}$ with $\mathbf{F}$'s i-th column
10: $\quad \mathbf{q}, \boldsymbol{\alpha}, \nu \leftarrow$ solution of 6 & 7, considering $\tilde{\mathbf{F}}$(of size $m \times t$)
11: **end for**
12: **return** $\mathbf{q}$

$\frac{1}{\pi} - 1 \in \mathbb{R}_+^*$

$$\min_{\pi \in]0,1[} \left(1 - \frac{\left(\sum_{i=1}^m [y_i(\pi f(x_i) + (1-\pi)g_i)]\right)^2}{\sum_{i=1}^m [y_i(\pi f(x_i) + (1-\pi)g_i)]^2} \frac{1}{m}\right)$$

$$= \min_{\beta \in \mathbb{R}_+^*} \left(1 - \frac{\frac{1}{(1+\beta)^2}\left(\sum_{i=1}^m [y_i(f(x_i) + \beta g_i)]\right)^2}{\frac{1}{(1+\beta)^2}\sum_{i=1}^m [y_i(f(x_i) + \beta g_i)]^2} \frac{1}{m}\right)$$

$$= \min_{\beta \in \mathbb{R}_+^*} \left(1 - \frac{\left(\sum_{i=1}^m [y_i(f(x_i) + \beta g_i)]\right)^2}{\sum_{i=1}^m [y_i(f(x_i) + \beta g_i)]^2} \frac{1}{m}\right)$$

□

Proof. **Proof of Theorem 3** In order to prove this, we define,

$$N_{f,g}(\beta) = \left(\sum_{i=1}^m [y_i\,(f(x_i) + \beta g(x_i))]\right)^2$$

and

$$D_{f,g}(\beta) = \sum_{i=1}^m \left(y_i\,[f(x_i) + \beta g(x_i)]\right)^2$$

Remark 1. *$N_{f,g}$ and $D_{f,g}$ are both quadratic functions in β.*

Let

$$N_{f,g}(\beta) = A_0 + A_1\beta + A_2\beta^2$$

and

$$D_{f,g}(\beta) = B_0 + B_1\beta + B_2\beta^2$$

In order to minimize $\Phi_{f,g}$, we have to derive it,

$$\Phi'_{f,g}(\beta) = -\frac{1}{m}\frac{N'_{f,g}(\beta)D_{f,g}(\beta) - N_{f,g}(\beta)D'_{f,g}(\beta)}{D_{f,g}(\beta)^2}$$

With

$$\begin{aligned} N'_{f,g} =& A_1 + 2A_2\beta \\ D'_{f,g} =& B_1 + 2B_2\beta \end{aligned}$$

So,

$$\begin{aligned} N'_{f,g}D_{f,g} =& A_1B_0 + \beta(2A_2B_0 + A_1B_1) \\ &+ \beta^2(2A_2B_1 + A_1B_2) + \beta^3 2A_2B_2 \\ D'_{f,g}N_{f,g} =& A_0B_1 + \beta(2A_0B_2 + A_1B_1) \\ &+ \beta^2(2A_1B_2 + A_2B_1) + \beta^3 2A_2B_2 \end{aligned}$$

To conclude,

$$\Phi'_{f,g}(\beta) = -\frac{1}{m}\frac{C_0 + \beta C_1 + \beta^2 C_2}{D_{f,g}(\beta)^2}$$

With,

$C_0 = A_1B_0$; $C_1 = 2(A_0B_2 - A_2B_0)$; $C_2 = A_1B_2 - A_2B_1$

So, to find the minimum, we need to zero the derivative function. To do that, we have to zero $C_0 + \beta C_1 + \beta^2 C_2$ and make sure that the solution will not zero $D_{f,g}$.

The roots of $C_0 + \beta C_1 + \beta^2 C_2$ are $\frac{-C_1 \pm \sqrt{C_1^2 + 4C_0C_2}}{2C_2}$ or $\frac{-C_1}{2C_2}$

Moreover, $D_{f,g} = B_0 + B_1\beta + \beta^2 B_2 = \|F\|_2^2 + 2m\tau(F,g)\beta + m\beta^2$.

So it's determinant is $4m(m\tau(F,g)^2 - \|F\|_2^2)$. Yet, we have assumed that $\tau(F,g) < 1$ and $\|F\|_2^2 > m$ so $m\tau(F,g)^2 < \|F\|_2^2$ so the determinant is negative, hence there is no real solution to $D_{f,g}(\beta) = 0$

So we have an analytic solution of the two-voters problem,

$$\min_{\beta \in \mathbb{R}_+^*} \Phi_{f,g}(\beta) = \begin{cases} \frac{-C_1 \pm \sqrt{C_1^2 + 4C_0C_2}}{2C_2} \\ \frac{-C_1}{2C_2} \end{cases} \tag{8}$$

□

Proof. **Proof of Property 2** Let's first develop C_0, C_1, C_2 from Theorem 3 as functions of $\gamma(g), \gamma(F), \tau(F,g), \|F\|_2^2$

$$\begin{aligned} C_2 &= 2m^3\gamma(g)\left(\gamma(F) - \gamma(g)\tau(F,g)\right) \\ C_1 &= 2m^2\left(m\gamma(F)^2 - \gamma(g)^2\|F\|_2^2\right) \\ C_0 &= 2m^2\gamma(F)\left(m\gamma(F)\tau(F,g) - \gamma(g)\|F\|_2^2\right) \end{aligned} \tag{9}$$

As we have seen in Equation 8, the weight is either $\frac{-C_1 \pm \sqrt{C_1^2 + 4C_0C_2}}{2C_2}$ or $\frac{-C_1}{2C_2}$.

— If $\beta = \frac{-C_1}{2C_2}$, it violates $\tau(F,g) < 1$ and $\|F\|_2^2 > m$. It means the determinant of $C_2\beta + C_1\pi + C_0$ is null. Which means, thanks to Equation 9 that

$$\begin{aligned} &\|F\|_2^2\gamma(g)^2 = \gamma(F)m\left(2\gamma(g)\tau(F,g) - \gamma(F)\right) \\ \Leftrightarrow\ &\|F\|_2^2\gamma(g)^2 - 2\gamma(F)m\gamma(g)\tau(F,g) - m\gamma(F)^2 = 0 \end{aligned} \tag{10}$$

The determinant of this quadratic problem in $\gamma(g)$ is

$$4\gamma(F)^2 m(m\tau(F,g)^2 - \|F\|_2^2)$$

Yet, we have assumed that $\tau(F,g) < 1$ and $\|F\|_2^2 > m$ so $m\tau(F,g)^2 < \|F\|_2^2$ so the determinant is negative, hence there is no real solution for 10.

Hence, $\beta \neq \frac{-C_1}{2C_2}$.

— Let's consider the option $\beta = \frac{-C_1 - \sqrt{C_1^2 + 4C_0C_2}}{2C_2}$. According to the assumption made in Equation 2, $C_1 = 2m^2\left(m\gamma(F)^2 - \gamma(g)^2\|F\|_2^2\right) > 0$ and $C_2 = 2m^3\gamma(g)\left(\gamma(F) - \gamma(g)\tau(F,g)\right) > 0$. So $-C_1 - \sqrt{C_1^2 + 4C_0C_2} < 0$ and $C_2 > 0$. Which is absurd as $\beta > 0$

In conclusion, $\beta = \frac{-C_1 + \sqrt{C_1^2 + 4C_0C_2}}{2C_2}$ □

Proof. **Proof of Property 1**

— In Property 1, we first stated that : $1 > \gamma(g) \geq \frac{1}{m}$. Indeed, as g is supposed to be a weak classifier, it is slightly better than the random classifier of null margin. As g outputs are in $\{-1, +1\}$, the smallest possible positive margin for it is to classify $\lfloor \frac{m}{2} \rfloor + 1$ examples well. So it's margin will be $\gamma(g) = \frac{1}{m}$. So, $\gamma(g) \geq \frac{1}{m}$.
Moreover, as g is a weak classifier, it won't perfectly classify all the examples, so it is safe so state $1 > \gamma(g)$

— We also stated that $\gamma(g) < \gamma(F)$ Let's suppose $\gamma(g) \geq \gamma(F)$, so, if we suppose F as been built over t iterations, we have $F = \sum_{j=1}^{t} F_j\beta_j$, with $\beta_0 = 1$ and $\beta_j \in \mathbb{R}_+^*$, $j = 2 \,..\, t$.

$$\begin{aligned} &\frac{1}{m}\sum_{i=1}^{m} F(x_i)y_i \leq \frac{1}{m}\sum_{i=1}^{m} g(x_i)y_i \\ \Rightarrow &\sum_{i=1}^{m}\sum_{j=1}^{t} \beta_j F_j(x_i)y_i \leq \sum_{i=1}^{m} g(x_i)y_i \\ \Rightarrow &\sum_{j=2}^{t}\sum_{i=1}^{m} \beta_j F_j(x_i)y_i + \sum_{i=1}^{m} F_0(x_i)y_i \leq \sum_{i=1}^{m} g(x_i)y_i \end{aligned}$$

And as F_0 has been chosen as the weak classifier with the largest margin, we have $\sum_{i=1}^{m} F_0(x_i)y_i \geq \sum_{i=1}^{m} g(x_i)y_i$

So : $\sum_{j=2}^{t}\sum_{i=1}^{m} \beta_j F_j(x_i)y_i \leq 0$ which is impossible because

— $\beta_j \in \mathbb{R}_+^*$, $j = 2 \,..\, t$,

— $\gamma(F_j) > \frac{1}{m}$ for the same reason as earlier.

So : $\gamma(g) < \gamma(F)$ □

Proof. **Proof of Property 4** Let's analyze $\sqrt{C_1^2 - 4C_0C_2}$:

$$\begin{aligned}C_1^2 - 4C_0C_2 =& 4m^4(m^2\gamma(F)^2\left(\gamma(F) - 2\gamma(g)\tau(F,g)\right)^2 \\ &+ \|F\|_2^2\left[2m\gamma(F)\gamma(g)^2\left(\gamma(F) - 2\gamma(g)\tau(F,g)\right)\right] \\ &+ \gamma(g)^4\|F\|_2^2)\end{aligned}$$

The determinant of this quadratic Equation in $\|F\|_2^2$ is,

$$\begin{aligned}&\left[2\gamma(F)\gamma(g)^2 m\left(\gamma(F) - 2\gamma(g)\tau(F,g)\right)\right]^2 \\ &-4\gamma(g)^4\gamma(F)^2m^2\left(\gamma(F) - 2\gamma(g)\tau(F,g)\right)^2 = 0\end{aligned}$$

So the double root is,

$$-\frac{\gamma(F)m\left(\gamma(F) - 2\gamma(g)\tau(F,g)\right)}{\gamma(g)^2}$$

So,

$$\begin{aligned}C_1^2 - 4C_0C_2 =& 4m^4\gamma(g)^4\Big(\|F\|_2^2 \\ &+\frac{\gamma(F)m\left(\gamma(F) - 2\gamma(g)\tau(F,g)\right)}{\gamma(g)^2}\Big)^2\end{aligned}$$

So thanks to assumption 3, we obtain Equation 11, at the end of this page.

Let's now analyze the $\mathcal{C}$-bound at iteration $t+1$,

$$\mathscr{C}_{t+1}^F = 1 - \frac{\left(\sum_{i=1}^m[y_i(F_i + \beta_{t+1}g_i)]\right)^2}{\sum_{i=1}^m[y_i(F_i + \beta_{t+1}g_i)]^2}\frac{1}{m}$$

And at iteration t,

$$\mathscr{C}_t^F = 1 - \frac{\left(\sum_{i=1}^m[y_iF_i]\right)^2}{\sum_{i=1}^m F_i^2}\frac{1}{m}$$

So the numerators and denominators are,

$$\begin{aligned}N_{t+1} =& \left(\sum_{i=1}^m[y_i(F_i + \beta_{t+1}g_i)]\right)^2 \\ D_{t+1} =& \sum_{i=1}^m[F_i + \beta_{t+1}g_i]^2\end{aligned}$$

so for iteration t,

$$\begin{aligned}N_t =& \left(\sum_{i=1}^m[y_iF_i]\right)^2 \\ D_t =& \sum_{i=1}^m[F_i]^2\end{aligned}$$

So we obtain $\mathscr{C}_{t+1}^F = \frac{N_{t+1}}{D_{t+1}} = \frac{N_t+c}{D_t+c'}$. Yet, to be able to quantify the $\mathcal{C}$-bound's decrease, we need to focus on $m(\mathscr{C}_t^F - \mathscr{C}_{t+1}^F) = \frac{N_t}{D_t} - \frac{N_t+c}{D_t+c'} = \frac{N_tc' - D_tc}{(D_t+c')D_t}$.

$$\begin{aligned}&C_1^2 - 4C_0C_2 = 4m^4\gamma(g)^4\left(\|F\|_2^2 + \frac{\gamma(F)m\left(\gamma(F) - 2\gamma(g)\tau(F,g)\right)}{\gamma(g)^2}\right)^2 \\ \Leftrightarrow\ &\sqrt{C_1^2 - 4C_0C_2} = 2m^2\gamma(g)^2\left(\|F\|_2^2 + \frac{\gamma(F)m\left(\gamma(F) - 2\gamma(g)\tau(F,g)\right)}{\gamma(g)^2}\right) \\ \Leftrightarrow\ &-C1 + \sqrt{C_1^2 - 4C_0C_2} = 4m^2\gamma(g)\left(\gamma(g)\|F\|_2^2 - \gamma(F)\tau(F,g)m\right) \\ \Leftrightarrow\ &\beta = \frac{-C1 + \sqrt{C_1^2 - 4C_0C_2}}{2C_2} = \frac{\gamma(g)\|F\|_2^2 - \gamma(F)\tau(F,g)m}{m\left(\gamma(F) - \gamma(g)\tau(F,g)\right)}\end{aligned} \tag{11}$$

Yet,

$$\begin{aligned}c =& 2\beta_{t+1}m^2\gamma(F)\gamma(g) + m^2\beta^2\gamma(g)^2 \\ c' =& 2\beta_{t+1}m\tau(F,g) + \beta^2m \\ N_t =& m^2\gamma(F)^2 \\ D_t =& \|F\|_2^2\end{aligned}$$

So,

$$\begin{aligned}D_tc =& \|F\|_2^2 m^2\gamma(g)\beta\left(2\gamma(F) + \beta\gamma(g)\right) \\ N_tc' =& m^3\gamma(F)^2\beta\left(2\tau(F,g) + \beta\right)\end{aligned}$$

We subtract,

$$\begin{aligned}N_tc' - D_tc = -\beta m^2\Big(&\beta\gamma(g)^2\|F\|_2^2 - \beta\gamma(F)^2m \\ &-2\gamma(F)^2m\tau(F,g) + 2\gamma(F)\gamma(g)\|F\|_2^2\Big)\end{aligned}$$

So, combining with β's expression found in Equation 11,

$$N_t c' - D_t c = -\frac{\left(\gamma(g)\left\|F\right\|_2^2 - \gamma(F)m\tau(F,g)\right)^2}{\left(\gamma(F) - \gamma(g)\tau(F,g)\right)^2} \times \left(\gamma(F)^2 m - 2\gamma(F)\gamma(g)m\tau(F,g) + \gamma(g)^2\left\|F\right\|_2^2\right)$$

Similarly,

$$\begin{aligned}(D_t + c')\,D_t &= \left\|F\right\|_2^2\left(\beta^2 2m + 2\beta m\tau(F,g) + \left\|F\right\|_2^2\right)\\ &= \frac{\left\|F\right\|_2^2\left(\left\|F\right\|_2^2 - m\tau(F,g)^2\right)}{\left(\gamma(F) - \gamma(g)\tau(F,g)\right)^2}\\ &\quad\times\left(\gamma(F)^2 m - 2\gamma(F)\gamma(g)m\tau(F,g) + \gamma(g)^2\left\|F\right\|_2^2\right)\end{aligned}$$

So,

$$\begin{aligned}\frac{N_t c' - D_t c}{(D_t + c')\,D_t} &= -\frac{\beta m^2\left(\beta\gamma(g)^2\left\|F\right\|_2^2 - \beta\gamma(F)^2 m - 2\gamma(F)^2 m\tau(F,g) + 2\gamma(F)\gamma(g)\left\|F\right\|_2^2\right)}{\left\|F\right\|_2^2\left(\beta^2 2m + 2\beta m\tau(F,g) + \left\|F\right\|_2^2\right)}\\ &= -\frac{m\left(\gamma(g)\left\|F\right\|_2^2 - \gamma(F)m\tau(F,g)\right)^2}{\left\|F\right\|_2^2\left(\left\|F\right\|_2^2 - m\tau(F,g)^2\right)}\end{aligned}$$

Then,

$$\begin{aligned}\mathscr{C}_t^F - \mathscr{C}_{t+1}^F &= \left(1 - \frac{N_t}{D_t}\frac{1}{m}\right) - \left(1 - \frac{N_t + c}{D_t + c'}\frac{1}{m}\right)\\ &= \left(\frac{N_t + c}{D_t + c'} - \frac{N_t}{D_t}\right)\frac{1}{m}\\ &= \left(\frac{D_t c - N_t c'}{(D_t + c)D_t}\right)\frac{1}{m}\\ &= \frac{\left(\gamma(g)\left\|F\right\|_2^2 - \gamma(F)m\tau(F,g)\right)^2}{\left\|F\right\|_2^2\left(\left\|F\right\|_2^2 - m\tau(F,g)^2\right)}\end{aligned}$$

In conclusion,

$$\mathscr{C}_t^F - \mathscr{C}_{t+1}^F = \frac{\left(\gamma(g)\left\|F\right\|_2^2 - \gamma(F)m\tau(F,g)\right)^2}{\left\|F\right\|_2^2\left(\left\|F\right\|_2^2 - m\tau(F,g)^2\right)}$$

□

Modèle LSTM encodeur-prédicteur pour la prévision court-terme de l'affluence dans les transports collectifs

Kevin Pasini[1,2], Mostepha Khouadjia[2], Allou Samé[1], Fabrice Ganansia[3], Patrice Aknin[2], and Latifa Oukhellou[1]

[1]Université Paris-Est, IFSTTAR, Cosys-Grettia, Champs-sur-Marne, France
[2]IRT SystemX, Paris-Saclay, France.
[3]SNCF- Innovation & Recherche, St Denis, France

Résumé

Les possibilités offertes en termes de collecte et de stockage de données permettent de renouveler les approches de modélisation dans le domaine du transport. L'exploitation croisée de différentes sources de données a pour vocation la création de services à forte valeur ajoutée pour l'usager. Les travaux détaillés dans cet article portent sur le développement de modèles de prévision à base de méthodes d'apprentissage notamment profond, pour la prévision court-terme de la charge (nombre de passagers) des trains. Cette prévision de l'affluence dans les trains peut servir à enrichir l'information voyageur à destination des usagers des transports collectifs qui peuvent ainsi mieux planifier leur déplacement. Elle peut également servir aux opérateurs de transport pour une régulation "à la demande" de l'offre de transport. La principale difficulté dans la prévision est liée à la variabilité intrinsèque des séries temporelles des charges à prédire, induite par l'influence de plusieurs paramètres dont ceux liés à l'exploitation (horaire, retard, type de mission. . .) et au contexte (information calendaire, grand évènement, météo,...). Nous proposons un modèle LSTM encodeur-prédicteur pour résoudre cette tâche de prévision. Plusieurs expérimentations sont menées sur des données réelles du réseau Transilien de la SNCF sur une durée d'un an et demi. Les résultats de prévision sont détaillés en vue de comparer les performances d'un tel modèle à plusieurs horizons temporels avec celles d'autres modèles plus classiques utilisés en prévision.

Mots-clef : Prévision, mobilité urbaine, séries temporelles, LSTM.

1 Introduction

La prévision de la demande de mobilité est un problème central dans l'organisation de tout système de transport. La capacité d'anticipation que confère un algorithme prédictif permet en effet de mieux planifier les déplacements et de spécifier à l'avance une offre adaptée, en s'aidant de critères usuels de type temps de parcours, ou de niveau de confort lié à l'affluence. A l'heure actuelle, l'offre pour les transports collectifs urbains fait déjà l'objet d'une adaptation à l'aide de méthodes de prévision de la demande, mais ces adaptations ont lieu longtemps en avance, en fonction d'informations calendaires, contextuelles et de fréquentations types estimées à partir d'enquêtes et de données de comptage collectés ponctuellement à des intervalles importants. L'introduction de nouvelles données temps réel ou quasi temps réel permet un suivi plus dynamique de la demande et elle ne fait qu'accroître l'intérêt pour des méthodes de prévision adaptées à ces nouvelles données.

Dans cet article, nous visons la prévision court-terme des charges (le nombre de passagers) des trains dans les transports collectifs. L'horizon temporel de prévision est court-terme, à savoir à 15mn, 30mn ou 1h. Cette prévision exploite deux sources de données hétérogènes : les données de comptage de passagers et les données de localisation automatique des trains.

Une grande majorité des travaux menés dans le domaine de la prévision de la demande en transport porte sur la prévision d'affluence dans les gares ou de flux avec un niveau agrégé (toutes les 15mn, 30mn, ...). L'originalité des travaux présentés réside dans la prise en compte du plan de transport réalisé pour la prévision, lequel peut s'éloigner assez fortement du plan de transport nominal. De fait, les séries tempo-

relles de charge que nous souhaitons prévoir possèdent un pas d'échantillonage temporel variable, qui dépend de l'exploitation réelle prenant ainsi en compte les retards, les dessertes et missions des trains.

Une revue de littérature des travaux sur le sujet de la prévision d'affluence ou de charge conduit à distinguer plusieurs méthodologies selon les types de données utilisées et les objectifs visés en terme applicatif. Les premiers travaux sur la prévision du niveau de congestion dans les transports collectifs ont été proposés par [CSC12] et [LC11]. En utilisant des techniques relativement simples basées sur des aggrégations historiques des données billettiques, les auteurs de [CSC12] ont proposé des modèles de prévision du niveau de congestion dans les transports publics. Dans l'article [LC11], les auteurs se sont intéressés à la recommandation tarifaire en y intégrant une étape de prévision des habitudes de déplacement. Ces dernières années, plusieurs travaux ont été menés sur le développement de modèles de prévision à base de méthodes d'apprentissage automatique. On citera les travaux de [TCEMO16] où un réseau de neurone récurrent de type LSTM a été proposé pour la prévision court-terme de flux de passagers dans un réseau de transport. D'autres modèles plus complexes tenant compte des liens spatio-temporels entre les séries temporelles à prédire ont été proposés dans [KZYC17], [DWML16] et [ZDDG17]. Plus récemment, on notera les travaux de [HODL$^+$18] qui s'attachent à la prévision de charges dans les tramways. Le problème est formalisé comme une tâche de classification où les différentes classes à inférer sont directement déduites du taux d'occupation des sièges dans les transports.

Dans cet article, la tâche de prévision est vue comme un problème de prévision à plusieurs horizons temporels sur des séries temporelles irrégulières impactées par plusieurs facteurs contextuels. Afin de prendre en compte ces spécificités et en s'appuyant sur les capacités d'abstraction des réseaux de neurones combinés à l'apprentissage de représentation [BCV13], nous proposons un modèle LSTM encodeur-prédicteur combinée avec un apprentissage de représentation sur les facteurs contextuels. Le but est de fournir une prévision de la charge des trains au départ d'une station sur plusieurs pas de temps, tenant compte des valeurs de charges passées et de l'ensemble des facteurs contextuels caractérisant l'exploitation des trains.

L'article est organisé de la manière suivante : la section 2 décrit les données et les spécificités applicatives. la section 3 présente le modèle proposé pour la prévision tandis que la section 4 détaille les expérimentations menées sur des données réelles et les résultats obtenus en les comparant aux performances de plusieurs modèles pour la prévision à un et plusieurs pas de temps. La section 5 conclut cet article en fournissant des perspectives à ce travail.

2 Description des données

Les données que nous considérons sont collectées sur une ligne de chemin de fer desservant une cinquantaine de gares situées au nord de la banlieue parisienne. Environ 250 000 passagers par jour sont transportés sur cette ligne. L'ensemble des données couvre une période de 18 mois, allant de janvier 2015 à juin 2016, sur 40 stations exploitées pendant la journée de 5h00 à 2h00 du matin. La base de données inclut à la fois des informations sur les horaires réalisés et sur les données de comptage des passagers à l'embarquement et au débarquement de chaque train stationné à chaque gare. Ces sources de données hétérogènes enrichies d'informations de calendrier, permettent de reconstituer le nombre de passagers à bord de chaque train (la charge) au départ d'une gare.

Le principal objectif des travaux concerne la prévision de séries de charges univariées pour chaque gare. Afin d'illustrer l'allure temporelle des séries à prédire, la figure 1 présente deux profils de charge recueillies sur deux gares. Cette figure met en évidence un certain nombre de spécificités que le modèle de prévision devra intégrer, à savoir :

— Une période d'échantillonnage variable en raison des horaires de train et de l'exploitation ferroviaire. Chaque gare a sa propre fréquence de passage des trains.
— Un profil temporel spécifique de chaque série. Le profil est directement lié à l'usage de la station et en particulier à sa localisation spatiale et aux caractéristiques géographiques de la zone urbaine qui l'entoure (densité de population, densité d'emploi, loisirs, etc.).
— Les séries de charge de train sont influencées par des facteurs de calendrier tels que le type de jour (jour de semaine ou week-end), jour férié, vacances scolaires, etc.
— Les séries de charge de train sont également impactées par les caractéristiques des trains et par leurs missions (ligne à destinations multiples, services ferroviaires variés).

Outre ces facteurs contextuels, la demande en transports en commun est également impactée par des événements (sociaux, culturels, sportifs, etc.). Le modèle de prévision pour chaque gare doit faire face

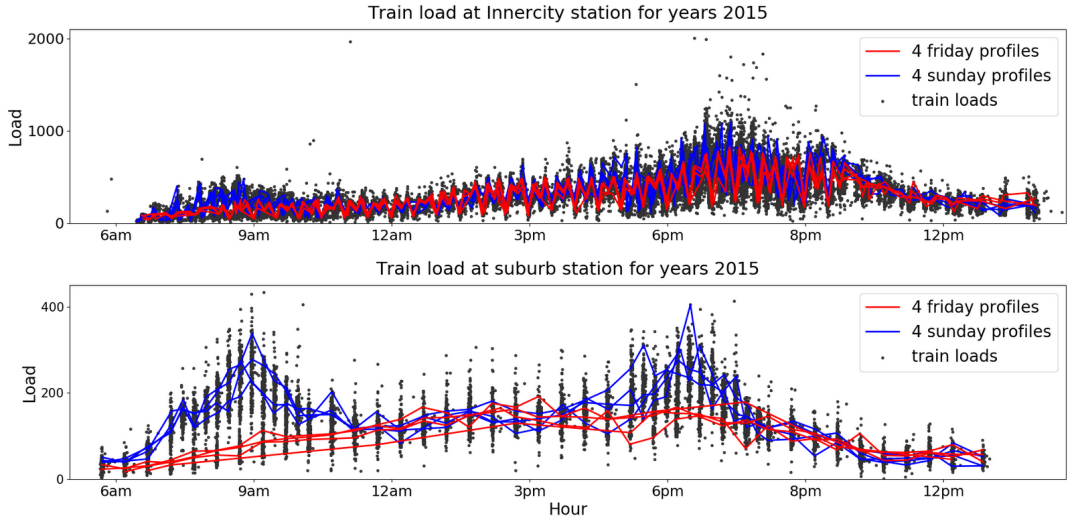


FIGURE 1 – Evolution des charges dans les trains sur l'année 2015 dans deux stations différentes

à tous ces facteurs temporels, spatiaux et exogènes énumérés ci-dessus. La section suivante détaille la méthodologie développée pour réaliser la tâche de prévision.

3 Modélisation proposée

Nous proposons un modèle dédié à la prévision sur plusieurs horizons temporels des séries de charges avec les spécificités précitées. Ce modèle prend la forme d'un réseau de neurones récurrent auquel on associe un apprentissage de représentation du contexte, ce qui permet de capter les différents facteurs contextuels pouvant influencer la prévision. La Table 1 résume les notations utilisées dans cet article.

L'objectif est de fournir une prévision de la série temporelle $(y_1,...,y_t)$ associée à une séquence d'observations S_t. Chaque élément de cette séquence est caractérisé par l'ensemble d'attributs contextuels e_t et de mesures passées m_t. On utilise la notation y_I, S_I, e_I pour désigner les sous-séquences $(y_t)_{t\in I}, (S_t)_{t\in I}, (e_t)_{t\in I}, (m_t)_{t\in I}$ avec $I \subset [1;T]$. Étant donné une fenêtre temporelle $W_i = [i-k, i+k']$ composée d'un horizon passé $P_i = [i-k, i[$, et d'un horizon futur $F_i = [i, i+k']$, le but de notre approche de prévision est d'inférer les réalisations y_{F_i} sur l'horizon futur à partir des informations disponibles sur la séquence d'observations $S_{W_i} = (e_{P_i}, e_{F_i}, m_{P_I})$ comme le montre la Figure 2. Le modèle doit être capable de capturer l'influence et les interactions des différents facteurs contextuelles sur les données historiques pour inférer au mieux la dynamique future de la série (y_t).

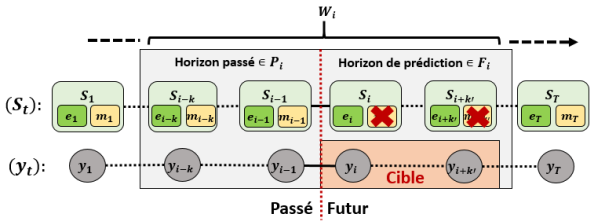


FIGURE 2 – Schéma général du modèle de prévision

S'inspirant des travaux de recherche sur les réseaux de neurones récurrents encodeur-décodeur proposés dans [CVMG+17], nous proposons un modèle de réseau de neurones LSTM encodeur-prédicteur (LSTP EP) pour la prévision court-terme et multi-horizons incluant un apprentissage de représentation des facteurs contextuels. A partir d'observations sur une fenêtre temporelle S_{W_i}, le modèle reconstruit les k dernières réalisations $\hat{y}_{P_i}$ et prévoit les k' prochaines réalisations $\hat{y}_{F_i}$ en exploitant les informations contextuelles passées et futures e_{P_i}, e_{F_i} ainsi que les informations de mesure

TABLE 1 – Notation et variable

Notation	
t	Horizon temporel $t \in [1, T]$
$y_1, ..., y_T$	(y_t) Séries de réalisation
$S_1, ..., S_t$	(S_t) Séquence d'observations
$e_1, ..., e_T$	(e_t) Séquence d'attributs contextuels
$m_1, ..., m_T$	(m_t) Séquence d'attributs de mesures
	Fenêtre
W_i	$[i-k, i+k']$: $i$ème Fenêtre temporelle.
P_i	$[i-k, i[$: Horizon passé de W_i
F_i	$[i, i+k']$: Horizon futur de W_i
X_i	$(m_{P_i}, e_{P_i}, e_{F_i})$ Attributs d'entrée
	Espace latent
$u_1, ..., u_T$	(u_t) Représentation contextuelles
$h_1, ..., h_T$	(h_t) Dynamique latente passée
$r_1, ..., r_T$	(r_t) État latent de reconstruction
$z_1, ..., z_T$	(z_t) État latent de prévision
	Modèle et ses composants
$LSTM_{EP}$	Modèle de réseau de neurones
$Fact$	Modèle contextuel
Enc	Encodeur récurrent
Dec	Décodeur récurrent
$Pred$	Prédicteur récurrent
$Reconst$	Sorties de reconstruction
$Predict$	Sorties de prévision

sur l'horizon passé.

$$\begin{aligned} LSTM_{EP}(X_i) &= LSTM_{EP}(m_{P_i}, e_{P_i}, e_{F_i}) \\ &= (\hat{y}_{P_i}, \hat{y}_{F_i}) \end{aligned} \tag{1}$$

Ce modèle peut être vu comme un réseau de neurones profond pouvant se décomposer en plusieurs éléments ayant un rôle spécifique. Une illustration générale est fournie dans la Figure 3.

Les composants du réseau de neurones sont décrits comme suit :

Fact : un modèle de contexte dédié à synthétiser les caractéristiques (e_t) en une représentation contextuelle (u_t). Il s'agit d'un pré-traitement basé sur un perceptron multi-couches appliqué à chaque observation afin de régulariser les représentations contextuelles.

$$Fact(e_{P_i}, e_{F_i}) = \bigoplus_{t \in (P_i \cup F_i)} Fact(e_t) = \bigoplus_{t \in (P_i \cup F_i)} u_t = (u_{P_i}, u_{F_i}) \tag{2}$$

Enc : un LSTM encodeur de type "many-to-one" dédié à la capture de la dynamique latente passée (h_i)

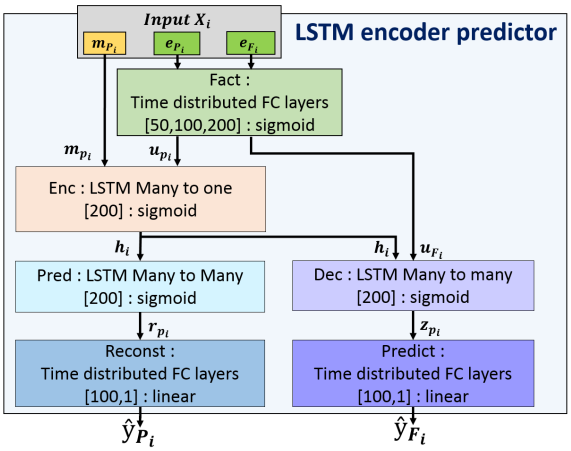


FIGURE 3 – Architecture générale du réseau LSTM encodeur-prédicteur

à partir des mesures m_{P_i} et des représentations contextuelles (u_{P_i}) de l'horizon passé.

$$Enc(m_{P_i}, u_{P_i}) = h_i \tag{3}$$

Dec : un LSTM décodeur de type "many-to-many" décodant récursivement les états latents de reconstruction r_{P_i} des observations passées à partir de la dynamique latente de l'horizon passé (h_i). Chaque état latent de reconstruction est ensuite interprété par des couches de reconstruction linéaire **'Reconst'** qui infèrent la réalisation des observations sur l'horizon passé.

A partir des sorties de **'Reconst'**, on obtient $\hat{y}_{P_i}$ qui est utilisé comme objectif intermédiaire durant la phase d'apprentissage pour faciliter la capture des dynamiques passées.

$$Dec(h_i) = r_{P_i} \tag{4}$$

$$Reconst(r_{P_i}) = \bigoplus_{t \in P_i} Reconst(r_t) = \hat{y}_{P_i} \tag{5}$$

Enc et *Dec* forment une structure encodeur-décodeur visant à synthétiser les dynamiques sur l'horizon passé à partir des attributs contextuels et des mesures sur cet horizon.

Pred : un LSTM prédicteur de type "many-to-many" inférant récursivement les états latents de prévision (z_{F_i}) des futures observations à partir de

leurs représentations contextuelles (u_{F_i}) en prenant en compte la dynamique latente sur l'horizon passé (h_i). Chaque état latent de prévision est ensuite interprété par des couches de reconstruction linéaire **'Predict'** pour inférer la réalisation de l'observation. A partir des sorties de 'Predict', on obtient $\hat{y}_{F_i}$ qui correspond à l'objectif de prévision multi-horizons.

$$Pred(h_i, u_{F_i}) = z_{F_i} \tag{6}$$

$$Predict(z_{F_i}) = \bigoplus_{t \in F_i} Predict(z_t) = \hat{y}_{F_i} \tag{7}$$

Notons que le modèle est intrinsèquement conçu pour prendre en compte des pas d'échantillonage variables, ce qui le rend robuste à la suppression d'observations (données manquantes).

3.1 Apprentissage des hyper-paramètres

Le réseau de neurone profond est entraîné à travers une rétropropagation du gradient minimisant la fonction de coût suivante :

$$\begin{gathered}\mathcal{L}(\theta) = \alpha_p * \sum_{t \in P_i} ||y_t - \hat{y}_t||^2 + \alpha_f * \sum_{t \in F_i} ||y_t - \hat{y}_t||^2 \\ \theta = (\theta_{Fact}, \theta_{Enc}, \theta_{Dec}, \theta_{Pred}, \theta_{Reconst}, \theta_{Predict}).\end{gathered} \tag{8}$$

Le premier terme mesure la capacité du modèle à reconstruire les observations antérieures à partir de la dynamique latente du passé. C'est un objectif intermédiaire qui vise à faciliter l'apprentissage des dynamiques passées. Le second terme mesure la capacité de prévision du modèle. Les hyper-paramètres α_p et α_f pondèrent les objectifs de reconstruction et de prévision.

Pour l'apprentissage, nous réalisons une optimisation par mini-batch grâce à la technique d'optimisation *Nadam* [SMDH13]. Les deux gradients de prévision et de reconstruction sont propagés depuis leurs couches de sorties (Predict and Reconst) vers les couches en amont en passant par le modèle de contexte à travers les couches du LSTM. Le modèle LSTM encodeur-prédicteur est implémenté dans l'environnement *TensorFlow* [ABC+16], et *Keras* [C+15].

Les hypere-paramètres ont été choisis empiriquement après plusieurs expériences sur la base des performances et de la convergence de l'apprentissage. *Fact* est composé de 3 couches denses de tailles [50, 100, 200] avec une fonction d'activation sigmoïde pour un total de 27000 paramètres. *Enc*, *Dec*, and *Pred* sont 3 couches LSTM de taille 200 avec une fonction d'activation sigmoïde pour un total de 880000 paramètres. *Reconst* et *Predict* sont composés de 2 couches denses de tailles [100, 1] avec une fonction d'activation linéaire pour un total de 40000 paramètres. Le réseau entier compte approximativement 900000 paramètres.

L'entraînement est réalisé de manière empirique avec des batchs de taille 128 sur plusieurs milliers d'itérations, ce qui prend quelques heures sur un GPU standard selon le jeu de données en entrée et la profondeur temporelle considérée. Des investigations complémentaires sont nécessaires pour accélérer la convergence du modèle.

4 Résultats expérimentaux et discussions

Pour la partie expérimentale, nous évaluons deux modèles de prévision de charge dans les trains sur la base de deux jeux de données relatifs respectivement à une station du centre-ville parisien, et une station de banlieue. Ces jeux de données couvrent une période d'exploitation comprise entre janvier 2015 et juin 2016, et sont répartis en un ensemble d'apprentissage représentant 66 % du jeu de données et un ensemble de test correspondant aux 33% restants. Les deux modèles utilisent plusieurs informations contextuelles "long-terme" de type calendaires :

- Jour de l'année : il correspond à la position du jour dans l'année (365 valeurs possibles encodées dans un vecteur de dimension 8 par l'application des fonctions cosinus et sinus sur 2x4 fréquences).
- Type de jour : il correspond à la position du jour dans la semaine et est encodé sur 8 dimensions avec une information supplémentaire si le jour est férié ou non.
- Minutes : Encodage des minutes journalières (1440 valeurs possibles) sur 8 dimensions par cosinus et sinus sur (2x4) fréquences.
- Service : caractéristique relative au service des trains en termes de mission et de branche desservie.

Par ailleurs, nous considérons également des caractéristiques sur un horizon court-terme en prenant en compte les observations antérieures sur une fenêtre couvrant les six derniers passages de trains à la station étudiée. Ces caractéristiques se résument à :

- Retard : Différence entre le temps théorique et le temps réel de passage du train à la station en minutes.
- Charge : Nombre de passagers dans le train pour

chacun des six derniers passages de trains à la station.
— Montants : Nombre de passagers qui sont montés à bord des trains lors des six derniers passages à la station.
— Descendants : Nombre de passagers qui sont descendus des trains lors des six deniers passages à la station.

4.1 Modèles de prévision

Dans cette section, les performances du modèle LSTM encodeur-prédicteur entraîné sur les deux catégories de caractéristiques e_t et m_t sont comparées avec celles d'autres modèles classiques issus de la littérature et qui vont servir de référence à cette comparaison. Ces modèles se résument en :

— **Dernière Valeur (DV)** : un modèle de prévision simple qui consiste à renvoyer la dernière charge observée dans le train comme étant la prochaine charge à prévoir à la même station.
— **Moyenne Contextuelle (MC)** : cette approche considère la charge moyenne constatée sur le même type de jour et la même tranche horaire comme étant la charge à prévoir.
— **Gradient Boosting (XGB)** [CG16] : modèle de régression appartenant à la famille des méthodes ensemblistes et qui regroupe des arbres de décision dits faibles. Deux modèles sont proposés selon la nature des caractéristiques employées en entrée du modèle :
  • **XGB LT** : désigne le modèle XGB entraîné sur la base des caractéristiques long-terme e_t.
  • **XGB ST** : désigne quant à lui le modèle XGB entraîné sur la base des caractéristiques court-terme. m_t.

Pour l'ajustement des hyper-paramètres des modèles XGB, une recherche par grille ("search-grid") est réalisée en utilisant une validation croisée avec k=3 (3-fold). Enfin, nous évaluons les performances des modèles sur chaque pas de temps correspondant au passage d'un train en utilisant comme mesure de performances l'erreur quadratique moyenne (RMSE) et le pourcentage d'erreur absolue pondérée (WAPE). Ce dernier peut être interprété comme le pourcentage de l'erreur totale comparée à la valeur moyenne de l'observation actuelle, et est calculé comme suit :

$$WAPE\ score : \frac{\sum_t ||y_t - \hat{y}_t||}{\bar{y}} \tag{9}$$

4.2 Évaluation

L'évaluation des modèles prédictifs est basée sur une comparaison des performances obtenues par ces derniers, et est exprimée à l'aide des métriques décrites dans la section précédente. Ces mesures sont calculées à la fois sur l'ensemble d'apprentissage et l'ensemble de test pour les deux stations étudiées, à savoir celle du centre ville et celle de banlieue. Les cinq modèles décrits dans la section 4.2 : Dernière valeur (DV), Moyenne Contextuelle (MC), le Gradient Boosting (XGB) dans ses deux variantes (ST) et (LT), ainsi que le LSTM EP sont comparés. Le Tableau 2 décrit les performances des différents modèles sur les stations étudiées.

Table 2 – Performances des modèles sur les stations étudiées

Modèle	Banlieue		Centre-ville	
	WAPE	RMSE	WAPE	RMSE
	Score d'apprentissage			
DV	17.9	35.8	41.9	186.7
MC	13.7	28.7	14.2	73.1
XGB LT	8.4	17.2	8.3	44.75
XGB ST	7.5	15.1	8.2	43.5
LSTM EP	10.7	22.1	10.9	57.7
	Score de Test			
DV	24.1	47.2	46.9	205.0
MC	19.0	40.0	18.5	96.5
XGB LT	18.8	38.9	13.4	76.0
XGB ST	16.8	35.7	**12.7**	73.0
LSTM EP	**16.0**	**33.8**	12.9	**72.4**

Comme attendu, les performances des modèles avancés (XGB, LSTM EP) surpassent celles des modèles DV et MC. Ces performances peuvent s'expliquer par le fait que les modèles XGB et LSTM EP ont une capacité de généralisation meilleure que les modèles de référence, ce qui permet d'éviter le problème de sur-apprentissage sur les données. Par ailleurs, les modèles XGB et LSTM EP sont plus complexes que les modèles de référence qui se contentent juste de renvoyer la dernière valeur observée ou la moyenne de la charge enregistrée sur un historique donné. Sur l'ensemble, le LSTM EP fournit de meilleurs résultats puisqu'à l'aide des caractéristiques court-terme, il est capable de capturer la dynamique du service des trains à la station.

En examinant l'erreur de prévision du LSTM EP sur la charge prédite (voir Figure 5), nous pouvons observer que le taux d'erreur croit avec l'augmentation de la charge à bord. Le modèle tend à surestimer légèrement les trains faiblement chargés, et à sous-

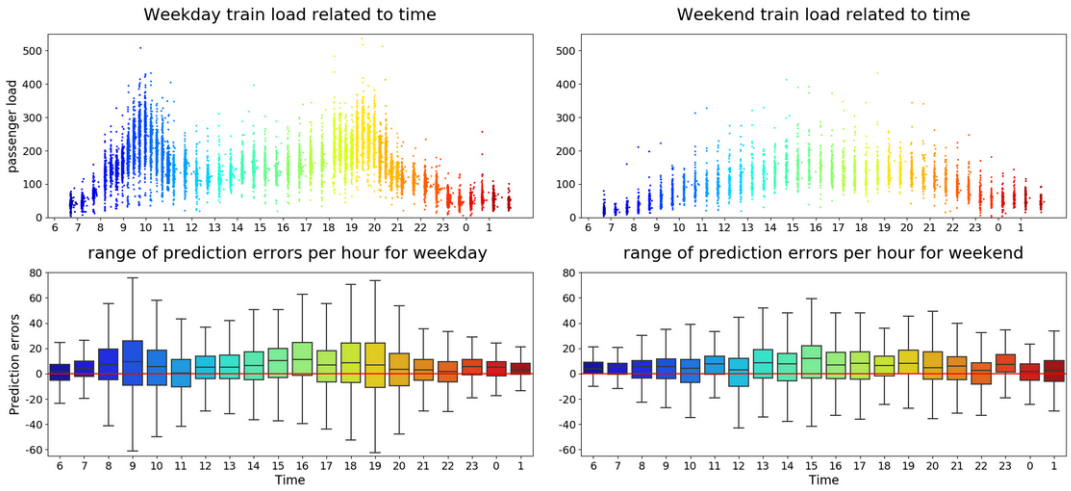


FIGURE 4 – L'erreur de prévision en fonction de la plage horaire pour la station de banlieue.

estimer les trains surchargés. Ceci peut être expliqué par le fait que les trains fortement chargés sont peu nombreux, et présentent des informations contextuelles similaires à celles des trains faiblement chargés, ce qui rend difficile la prévision des charges élevées.

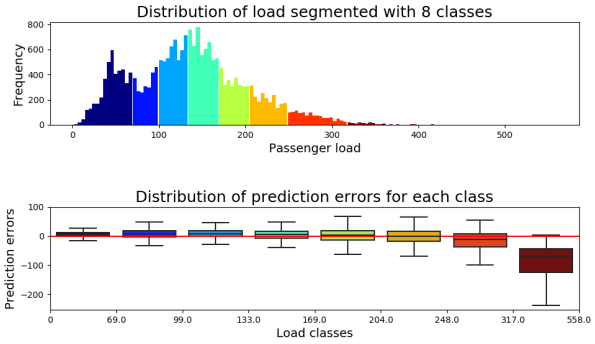


FIGURE 5 – L'erreur de prévision en fonction du segment de charge pour la station de banlieue.

Comme l'illustre la Figure 4, l'erreur commise par le modèle est du même ordre de grandeur pour les jours de semaine que pour les jours de week-end avec néanmoins quelques différences. La variance de l'erreur calculée sur les créneaux horaires de la journée s'avère corrélée avec la charge à bord. Sur les jours de semaine, on constate des erreurs importantes durant les heures de pointe en lien avec une variance importante et une charge élevée. Le modèle commet des erreurs moyennes en milieu de journée et des erreurs faibles tôt le matin et en fin de soirée. Par ailleurs, durant les week-ends excepté en matinée, on observe une variance de l'erreur relativement stable avec une valeur maximale atteint à midi et au milieu de l'après-midi. On note également que le modèle a plus de difficulté à prévoir les soirées de week-end que celles des jours de semaine.

D'autre part, lorsqu'on examine les performances des modèles de prévision sur des pas temporels successifs (multi-horizons), le LSTM EP surpasse XGB sur les six horizons (Tableau 3 et Tableau 4).

TABLE 3 – RMSE sur la station de banlieue pour les modèles de prévision multi-horizons.

Modèle	t+1	t+2	t+3	t+4	t+5	t+6
Time interval	14-32	29-62	44-92	59-122	75-152	90-182
XGB LT	38.9	38.9	38.9	38.9	38.9	38.9
XGB ST	35.7	36.6	36.7	36.7	37.6	38.1
LSTM EP	**33.8**	**34.0**	**34.1**	**34.4**	**34.7**	**34.9**

Ces horizons correspondent aux prochains passages des trains à la station et varient entre 14 et 182 minutes pour la station de banlieue, et entre 2 et 61 minutes pour la station du centre ville. Cette différence est due à une fréquence de trains plus élevée en centre ville qu'en banlieue. Il est important de souligner que ces prévisions sont obtenues avec un modèle LSTM EP unique qui prévoit simultanément la charge dans

Table 4 – RMSE sur la station du centre-ville pour les modèles de prévision multi-horizons.

Modèle	t+1	t+2	t+3	t+4	t+5	t+6
Time interval	2-13	5-23	9-31	12-43	15-53	18-61
XGB LT	76.0	76.0	76.0	76.0	76.0	76.0
XGB ST	73.0	72.8	73.3	73.8	73.4	73.5
LSTM EP	**72.4**	**72.1**	**72.1**	**72.2**	**72.6**	**72.8**

les trains sur l'ensemble des horizons temporels, tandis que pour le modèle XGB, nous avons autant de modèles que d'horizons temporels. Les prévisions fournies par le modèle XGB LT restent invariantes au fil des horizons temporels. Cependant, les performances de la variante XGB ST se dégradent avec l'avancement dans le temps. Le LSTM EP maintient des prévisions compétitives et robustes sur l'ensemble des horizons temporels aussi bien en station du centre-ville qu'en banlieue. Ceci peut être expliqué par une meilleure compréhension des facteurs contextuels à travers la représentation latente qui contribue à capturer la dynamique sous-jacente du service lié aux trains qui desservent la station.

5 Conclusion

Dans cet article, un modèle de prévision à base d'un réseau de neurones LSTM encodeur-prédicteur (LSTM EP) associé à un apprentissage de représentation des facteurs contextuels a été proposé. Ce modèle vise à prévoir la charge dans les trains à plusieurs horizons temporels en tenant compte des facteurs contextuels et des horaires réels de passage des trains dans les stations. Les difficultés posées par ce problème de prévision sont induites par une variabilité de l'exploitation ferroviaire et par plusieurs facteurs contextuels. Le réseau de neurones que nous proposons a la particularité de pouvoir apprendre une représentation à partir de caractéristiques contextuelles, de capturer la dynamique passée latente à travers la sous-structure sous-jacente de l'encodeur-décodeur, puis de prévoir la dynamique à venir à l'aide de la couche prédictive.

Au travers des résultats obtenus sur un jeu de données réelles couvrant une année et demi d'exploitation, nous avons pu montrer le potentiel du modèle proposé à traiter la prévision à court terme de la charge des trains en comparaison à d'autres modèles tels que le modèle gradient boosting. Les performances du modèle proposé ont été évaluées sur deux gares avec des profils temporels différents et pour des horizons de prévision à un et plusieurs pas de temps. Sur les deux configurations, le LSTM-EP présente une meilleure robustesse de la qualité des prévisions fournies.
Les futurs travaux devront s'attacher à l'exploration de la représentation apprise, et en particulier, la capacité de l'espace latent prédictif à caractériser des situations anormales telles que des perturbations et des anomalies de trafic.

Références

[ABC+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow : A system for large-scale machine learning. *OSDI Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning : A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1798–1828, 2013.

[C+15] François Chollet et al. Keras. 2015.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost : A scalable tree boosting system. *SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 785–794, 2016.

[CSC12] Irina Ceapa, Chris Smith, and Licia Capra. Avoiding the crowds : Understanding tube station congestion patterns from trip data. *ACM SIGKDD International Workshop on Urban Computing*, pages 134–141, 2012.

[CVMG+17] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP Empirical Methods in Natural Language Processing*, page 1724–1734, 2017.

[DWML16] Chuan Ding, Donggen Wang, Xiaolei Ma, and Haiying Li. Predicting short-term subway ridership and prioritizing its influential factors using gradient boosting decision trees. *Sustainability*, 8(11) :1100, 2016.

[HODL+18] Léonie Heydenrijk-Ottens, Viktoriya Degeler, Ding Luo, N van Oort, and JWC van Lint. Supervised learning : Predicting passenger load in public transport. 2018.

[KZYC17] Jintao Ke, Hongyu Zheng, Hai Yang, and Xiqun Michael Chen. Short-term forecasting of passenger demand under on-demand ride services : A spatio-temporal deep learning approach. *Transportation Research Part C : Emerging Technologies*, 85 :591–608, 2017.

[LC11] Neal Lathia and Licia Capra. Mining mobility data to minimise travellers' spending on public transport. *17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1181–1189, 2011.

[SMDH13] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. *ICML International Conference on Machine Learning*, 28(1139-1147) :5, 2013.

[TCEMO16] Florian Toqué, Etienne Côme, Mohamed Khalil El Mahrsi, and Latifa Oukhellou. Forecasting dynamic public transport origin-destination matrices with long-short term memory recurrent neural networks. *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1071–1076, 2016.

[ZDDG17] Ali Ziat, Edouard Delasalles, Ludovic Denoyer, and Patrick Gallinari. Spatio-temporal neural networks for space-time series forecasting and relations discovery. *IEEE International Conference on Data Mining ICDM*, pages 705–714, 2017.

Ensemblist Variational AutoEncoder: latent representation of semi-structured data

Victor Berger[1] and Michele Sebag[1]

[1]TAU, CNRS − INRIA − Univ. Paris-Saclay, France

Abstract

Conditional Generative Models are now acknowledged an essential tool in Machine Learning. This paper focuses on their control. While many approaches aim at disentangling the data through the coordinate-wise control of their latent representations, another direction is explored in this paper. The proposed EVAE handles data with a natural multi-ensemblist structure (*i.e.* that can naturally be decomposed into elements). Derived from Bayesian variational principles, EVAE learns a latent representation leveraging both observational and symbolic information. A first contribution of the approach is that this latent representation supports a compositional generative model, amenable to multi-ensemblist operations (addition or subtraction of elements in the composition). This compositional ability is enabled by the invariance and generality of the whole framework w.r.t. respectively, the order and number of the elements. The second contribution of the paper is a proof of concept on synthetic 1D and 2D problems, demonstrating the efficiency of the proposed approach.
Keywords: Generative model, semi-structured representation, neural networks

1 Introduction

Representation learning is at the core of machine learning, and even more so since the inception of deep learning [BCV13]. As shown by e.g., [CDH+16, HMP+17], the latent representations built to handle high-dimensional data can effectively support desirable functionnalities. One such functionality is the ability to directly control the observed data through the so-called representation disentanglement, especially in the context of computer vision and image processing [Pra12, LOW+18] (more in section 2).

This paper extends the notion of representation disentanglement from a latent coordinate-wise perspective to a semi-structured setting. Specifically, we tackle the ensemblist setting where a datapoint can naturally be interpreted as the combination of multiple parts. The contribution of the paper is a generative model built on the Variational Auto-Encoder principles [KW13, RMW14], *controlling the data generation from a description of its parts* and supporting ensemblist operations such as the addition or removal of any number of parts.

The applicative motivation for the presented approach, referred to as *Ensemblist Variational AutoEncoder* (EVAE), is the following. In the domain of Energy Management, a key issue is to simulate the consumption behavior of an ensemble of consumers, where each household consumption is viewed as an independent random variable following a distribution law defined from the household characteristics, and the household consumptions are possibly correlated through external factors such as the weather, or a football match on TV (attracting members of some but not all households). Our long term goal is to infer a simulator, taking as input the household profiles and their amounts: it should be able to simulate their overall energy consumption and account for their correlations. The data-driven inference of such a programmable simulator is a quite desirable alternative to the current approaches, based on Monte-Carlo processes and requiring either to explicitly model the correlations of the elementary random variables, or to proceed by rejection.

Formally, given the description of datapoints and their parts, the goal of EVAE is to learn the distribution laws of the parts (here, the households) and to sample the overall distribution defined from a varying

number of parts (the set of households), while accounting for the fact that the parts are not independent, and the sought overall distribution depends on shared external factors: *the whole is not the sum of its parts.*

The paper is organized as follows. Section 2 briefly reviews related work in the domain of generative models and latent space construction, replacing our contribution in context. Section 3 gives an overview of EVAE, extending the VAE framework to multi-ensemblist settings. Section 4 presents the experimental setting retained to establish a proof of concept of the approach on two synthetic problems, and section 5 reports on the results. Finally section 6 discusses some perspectives for further work and applications to larger problems.

2 Related work

Generative models, including VAEs [KW13, RMW14] and GANs [GPAM+14], rely on an embedding from the so-called latent space Z onto the dataspace X, endowed with a variational structure. In the following, data space and observed space are used interchangeably. It has long been observed that continuous or discrete operations in the latent space could be used to produce interesting patterns in the data space. For instance, the linear interpolation between two latent points z and z' can be used to generate a morphing between their images [RMC15], or the flip of a boolean coordinate of z can be used to add or remove an elementary pattern (the presence of glasses or moustache) in the z image [DBP+17].

The general question then is to control the flow of information from the latent to the observed space and to make it actionable. Several approaches, either based on information theory or on supervised learning have been proposed to do so. Losses inspired from the Information Bottleneck [TPB00, SZT17, AS18] and enforcing the independence of the latent and the observed variables, conditionally to the relevant content of information, have been proposed: enforcing the decorrelation of the latent coordinates in β-VAE [HMP+17]; aligning the covariances of latent and observed data in [KSB17]; decomposing the latent information into pure content and pure noise in InfoGAN [CDH+16]. Independently, explicit losses have been used to yield conditional distributions in conditional GANs [MO14], or to enforce the scope of a latent coordinate in [KWKT15, TYL17], (e.g. modelling the light orientation or the camera angle).

The structure of the observed space can be mimicked in the latent space, to afford expressive yet trainable model spaces; in Ladder-VAE [SRM+16], a sequence of dependent latent variables are encoded and reversely decoded to produce complex observed objects. Auxiliary losses are added in [MSSW16] in the spirit of semi-supervised learning. In [KRMW14], the overall generative model involves a classifier, trained both in a supervised way with labelled examples and in an unsupervised way in conjuction with a generative model.

An important case study is that of sequential structures: [CKD+15] considers fixed-length sequences and loosely mimicks an HMM process, where latent variable z_i controls the observed variable x_i and the next latent z_{i+1}. In [HZG17], a linear relation among latent variables z_i and z_{i+1} is enforced; in [CRLG+18], a recurrent neural net is used to produce the latent variable encoding the current situation. In a more general context, [WGZ+18] provides a generic method for designing an appropriate inference network that can be associated with a given Bayesian network representing a generative model to train.

The injection of explicit information at the latent level can be used to support "information surgery" via loss-driven information parcimony. For instance in the domain of signal generation [CWBO19], the neutrality of the latent representation w.r.t. the locutor identity is enforced by directly providing the identity at the latent level: as z does not need to encode the locutor information, the information parcimony pressure ensures z independence wrt the locutor. Likewise, *fair* generative processes can be enforced by directly providing the sensitive information at the latent level [ZWS+13]. In [LSL+15], an adversarial mechanism based on Maximum Mean Discrepancy [GBR+12] is used to enforce the neutrality of the latent. In [MGB+18], the minimization of the mutual information is used in lieu of an adversary.

Discussion. All above approaches (with the except of sequential settings [CKD+15, HZG17], see below) handle the generation of a datapoint as a whole (naturally involving diverse facets; but not composed of inter-related parts). Our goal is instead to tackle the proper parts-and-whole structure of a datapoint, where the *whole is not necessarily the simple sum of its parts* and the parts of the whole are interdependent. In sequential settings [CKD+15, HZG17], the dependency of the elements in the sequence are handled through parametric restrictions (respectively considering fixed sequence-size or linear temporal dependency) to enforce the proper match of the observed and latent spaces. A key contribution of the proposed EVAE

is to tackle the parts-to-whole structure with no such restrictions, and specifically accommodating a varying number of parts − possibly different between the training and the generation phases.

3 Overview of EVAE

This section describes the EVAE model, building upon the VAE principles [KW13] with the following difference: EVAE aims at building a *programmable generative model* p_θ, taking as input the ensemble of the parts of a whole observed datapoint. A key question concerns the latent structure most appropriate to reflect the ensemblist nature of the observed data. The proposed structure (section 3.1) involves a latent variable associated to each part of the whole. The aggregation of the part is achieved through an order-invariant operation, and the interactions among the parts are modelled at an upper layer of the latent representation.

In encoding mode, the structure is trained from the pairs formed by a whole, and an abstract description of its parts; the latent variables are extracted along an iterative non-recurrent process, oblivious of the order and number of the parts (section 3.2) and defining the encoder model q_ϕ.
In generative mode, the generative model is supplied with a set of parts, and accordingly generates a consistent whole, where variational effects operate jointly at the parts and at the whole levels.

Notations. A datapoint x is associated with an ensemble of parts noted $\{\ell_i\}$. Each ℓ_i belongs to a finite set of categories Λ. Elements and parts are used interchangeably in the following. In our illustrating example, a consumption curve x involves a number of households; the i-th household is associated with its consumer profile ℓ_i, with ℓ_i ranging in a finite set of profiles. Each profile in Λ thus occurs 0, 1 or several times. The generative model relies on a learned distribution $p_\theta(x|\{\ell_i\})$, that is decomposed into latent variables: a latent variable named w_i associated to each part ℓ_i, and a common latent variable z.

3.1 EVAE: Bayesian architecture

The architecture proposed for EVAE is depicted as a graphical model on Fig. 1. As said, the i-th part belongs to category ℓ_i and is associated with a latent variable w_i (different parts with same category are associated with different latent variables). The ensemble of the w_is is aggregated into an intermediate latent

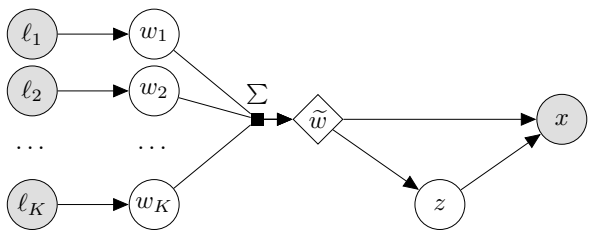


Figure 1: Bayesian network representation of the EVAE generative model.

variable $\widetilde{w}$. A key requirement is for $\widetilde{w}$ to be *invariant* w.r.t. the order of elements in x. In the following $\widetilde{w}$ is set to the sum of the w_i, $\widetilde{w} = \sum_i w_i$. Considering other order-invariant aggregations is left for further work.
The intermediate latent variable $\widetilde{w}$ is used to condition the z latent variable; both $\widetilde{w}$ and z condition the observed datapoint x. This scheme corresponds to the following factorization of the generative model p_θ:

$$p_\theta(x, z, \{w_i\}|\{\ell_i\}) = p_\theta(x|z, \widetilde{w})p_\theta(z|\widetilde{w}) \prod_i p_\theta(w_i|\ell_i) \tag{1}$$

In summary, the distribution of x is conditioned on the ensemble $\{\ell_i\}$ as follows: The i-th part of x is associated with a latent variable w_i modeling the generic distribution of the underlying category ℓ_i together with its specifics. Variable $\widetilde{w}$ is deterministically computed to model the aggregation of the w_i, and finally z models the specifics of the aggregation.

Notably, each w_i is linked to a single ℓ_i element, while z is global, being conditioned from the global auxiliary $\widetilde{w}$. The rationale for introducing z is to enable a more complex though still learnable distribution at the x level − compared with the alternative of conditioning x only on $\widetilde{w}$. It is conjectured that an information-effective distribution would store in w_i (respectively in z) the *local information* related to the i-th part (resp. the *global information* describing the interdependencies between all parts, e.g. the fact that the households face the same weather, vacation schedules, and so on). Along this line, it is conjectured that the extra information stored in z is limited compared to that stored in the w_is; we shall return to this point in section 4.1.

The property of invariance of the distribution w.r.t. the order of the ℓ_i is satisfied by design. A second desirable property regards the robustness of the distribution w.r.t. the varying number of parts in x. More precisely, two requirements are defined. The former one, referred

to as *size-flexibility property*, is that the number K of parts of an x is neither constant, nor bounded *a priori*. The latter one, referred to as *size-generality property* is the generative model p_θ to accommodate larger numbers of parts than those seen in the training set.

3.2 Posterior inference and loss

Letting $p_D(x|\{\ell_i\})$ denote the empirical data distribution, the learning criterion to optimize is the data likelihood according to the sought generative model p_θ: $\mathbb{E}_{p_D} \log p_\theta(x|\{\ell_i\})$.
The (intractable) posterior inference of the model is approximated using the Evidence Lower Bound (ELBO) [JGJS99], following the Variational AutoEncoder approach [KW13, RMW14]. Accordingly, we proceed by optimizing a lower bound of the log-likelihood of the data given p_θ, which is equivalent to minimizing an upper bound of the Kullback-Leibler divergence between the two distributions :

$$D_{KL}(p_D \| p_\theta) \leq H(p_D) + \mathop{\mathbb{E}}_{x \sim p_D} \mathcal{L}_{ELBO}(x) \tag{2}$$

The learning criterion reads, with $q_\phi(z, \{w_i\}|x, \{\ell_i\})$ the auxiliary inference distribution:

$$\begin{aligned} \mathcal{L}_{ELBO}(x) = & \mathop{\mathbb{E}}_{z,\{w_i\}\sim q_\phi} \log \frac{q_\phi(z, \{w_i\}|x, \{\ell_i\})}{p_\theta(z|\widetilde{w}) \prod_i p_\theta(w_i|\ell_i)} \\ & - \mathop{\mathbb{E}}_{z,\{w_i\}\sim q_\phi} \log p_\theta(x|z, \widetilde{w}) \end{aligned} \tag{3}$$

The inference distribution is further factorized as $q_\phi(\{w_i\}|z, x, \{l_i\}) q_\phi(z|x)$, yielding the final training loss:

$$\begin{aligned} \mathcal{L}_{ELBO}(x) = & \mathop{\mathbb{E}}_{z,\{w_i\}\sim q_\phi} \log \frac{q_\phi(\{w_i\}|x, z, \{\ell_i\})}{\prod_i p_\theta(w_i|\ell_i)} \\ & + \mathop{\mathbb{E}}_{z,\{w_i\}\sim q_\phi} \log \frac{q_\phi(z|x)}{p_\theta(z|\widetilde{w})} \\ & - \mathop{\mathbb{E}}_{z,\{w_i\}\sim q_\phi} \log p_\theta(x|z, \widetilde{w}) \end{aligned} \tag{4}$$

The training of the generative and encoder model distributions is described in Alg. 1.

3.3 Discussion

In EVAE, the sought distributions are structured as a Bayesian graph (see p_θ in Fig. 1), where each node is associated with a neural network and a probability distribution family, like for VAEs. This neural network

Algorithm 1 EVAE Training Procedure

$\theta, \phi \leftarrow$ Random initialization
while Not converged **do**
  $x, \{\ell_i\} \leftarrow$ Sample minibatch
  $z \leftarrow$ Sample from $q_\phi(z|x)$
  $\{w_i\} \leftarrow$ Sample from $q_\phi(\{w_i\}|x, z, \{\ell_i\})$
  $\mathcal{L}_w \leftarrow D_{KL}(q_\phi(\{w_i\}|x, z, \{\ell_i\}) \| \Pi_i p_\theta(w_i|\ell_i))$
  $\mathcal{L}_z \leftarrow \log \frac{q_\phi(z|x)}{p_\theta(z|\tilde{w})}$
  $\mathcal{L}_x \leftarrow -\log p_\theta(x|z, \tilde{w})$
  $\mathcal{L}_{ELBO} \leftarrow \mathcal{L}_w + \mathcal{L}_z + \mathcal{L}_x$
  $\theta \leftarrow \text{Update}(\theta, \nabla_\theta \mathcal{L}_{ELBO})$
  $\phi \leftarrow \text{Update}(\phi, \nabla_\phi \mathcal{L}_{ELBO})$
end while

takes as input the parent variables in the Bayesian graph, and outputs the parameters of a distribution in the chosen family, e.g., the mean and variance of a Gaussian distribution. The reparametrization trick [KW13] is used to back-propagate gradients through the sampling.

A concern regards the training of latent variables when considering Gaussian distributions. A potential source of instability in EVAE comes from the fact that the Kullback-Leibler divergence between q_ϕ and p_θ (Eq. (4)) becomes very large when the variance of some variables in p_θ becomes very small[1]. To limit this risk, some care is exercized in parameterizing the variances of the normal distributions in p_θ to making them lower-bounded.

3.3.1 Modelling of $q_\phi(\{w_i\}|x, z, \{\ell_i\})$.

The latent distributions $p_\theta(z|\widetilde{w})$, $p_\theta(w_i|\ell_i)$ and $q_\phi(z|x)$ are modelled using diagonal normal distributions as usual. Regarding the model $q_\phi(\{w_i\}|z, x, \{\ell_i\})$, in order to be able to faithfully reflect the generative model p_θ, it is necessary to introduce the correlation between the w_is in $q_\phi(\{w_i\}|z, x, \{\ell_i\})$ [WGZ+18].

As the aggregation of the w_i is handled by considering their sum, it is natural to handle their correlations through a multivariate normal distribution over the w_i. The proposed parametrization of such a multivariate is as follows. Firstly, correlations operate in a coordinate-wise fashion, that is, $w_{i,j}$ and $w_{i',j'}$ are only correlated if $j = j'$. The parametrization of the w_is ensures that: i) the variance of the sum of the $w_{i,j}$ can be controlled and made arbitrarily small in order to ensure an accurate VAE reconstruction; ii) the Kullback-Leibler divergence between $q_\phi(\{w_i\}|x, z, \{\ell_i\})$ and $\prod_i p_\theta(w_i|\ell_i)$

[1] Single-latent variable VAEs do not face such problems as the prior distribution $p_\theta(z)$ is fixed, it is not learned.

can be defined in closed form.

The learning of $q_\phi(\{w_i\}|x, z, \{\ell_i\})$ is done using a fully-connected graph neural network [SGT+09] leveraging graph interactions akin message-passing [GSR+17]. The graph has one node for each element ℓ_i, and every node is connected to all other nodes. The state of the i-th node is initialized to the concatenation of some learned function of x noted $pre_\phi(x)$, z, a learned embedding of ℓ_i noted $e_\phi(\ell_i)$ augmented with some random noise ϵ_i (to ensure the differentiation of the w_is). The state of each node of the graph at the k-th layer is then defined by its $k-1$-th layer state and the aggregation of the state of all other nodes:

$$\begin{cases} h_i^{(0)} = (pre_\phi(x), z, e_\phi(\ell_i) + \epsilon_i) \\ h_i^{(k)} = f_\phi^{(k)} \left(h_i^{(k-1)}, \sum_{j \neq i} g_\phi^{(k)}(h_j^{(k-1)}) \right) \end{cases} \quad (5)$$

where $f_\phi^{(k)}$ and $g_\phi^{(k)}$ are learned neural networks: $g_\phi^{(k)}$ is meant to embed the current state of each node for an aggregate summation, and $f_\phi^{(k)}$ is meant to "fine-tune" the i-th node conditionally to all other nodes, such that they altogether account for $\widetilde{w}$.

4 Experimental setting

This section presents the goals of experiments and describes the experimental setting used to empirically validate EVAE.

4.1 Goals of experiments

As said, EVAE is meant to achieve a programmable generative model. From a set of latent values w_i, either derived from $p_\theta(w_i|\ell_i)$ in a generative context, or recovered from some data x, it should be able to generate values $\hat{x}$ matching any chosen subset of the w_i. This property is what we name the "ensemblist disentanglement" capacity, and the first goal of these experiments is to investigate whether EVAE does have this capacity.

A second goal of these experiments is to examine whether the desired properties (section 3.1) hold. The order-invariant property is enforced by design. The size-flexibility property will be assessed by inspecting the sensitivity of the extraction and generative processes to the variability of the number of parts. The size-generality property will be assessed by inspecting the quality of the generative model when the number of parts increases significantly beyond the size range used during training.

A last goal is to understand how EVAE manages to store the information of the model in respectively the w_is and z. The conjecture done (section 3.1) was that the latent w_is would take in charge the information of the parts, while the latent z would model the interactions among the parts. The use of synthetic problems where the quantity of information required to encode the parts can be quantitatively assessed will permit to test this conjecture. A related question is whether the generative model is able to capture the fact that the whole is not the sum of its parts. This question is investigated using non-linear perturbations, possibly operating at the whole and at the parts levels, and comparing the whole perturbed x obtained from the ℓ_is, and the aggregation of the perturbed x_is generated from the ℓ_i parts. The existence of a difference, if any, will establish the value of the EVAE generative model compared to a simple Monte-Carlo simulator, independently sampling parts and thereafter aggregating them.

4.2 1D and 2D Proofs of concept

Two synthetic problems have been considered to empirically answer the above questions.[2]

In the 1D synthetic problem, the set Λ of categories is a finite set of frequencies $\lambda_1 \ldots \lambda_{10}$. A given "part" (here, curve) is a sine wave defined by its frequency ℓ_i in Λ and its intrinsic features, that is, its amplitude a_i and phase κ_i. The whole $x =_{def} \{\ell_1, \ldots \ell_K\}$ is a finite sequence of size T, deterministically defined from the non-linear combination of the curves:

$$x(t) = K \tanh \left(\frac{C}{K} \sum_{i=0}^{K} a_i \cos \left(\frac{2\pi \ell_i}{T} t + \kappa_i \right) \right)$$

with K the number of sine waves in x, C a parameter controlling the non-linearity of the aggregation of the curves in x, and T a global parameter controlling the sampling frequency. For each part (sine wave), a_i is sampled from $\mathcal{N}(1; 0.3)$, and κ_i is sampled from $\mathcal{N}\left(0; \frac{\pi}{2}\right)$.

The part-to-whole aggregation is illustrated on Fig. 2, plotting the non-linear transformation of the sum of 4 sine waves, compared to the sum of non-linear transformations of the same sine waves. C is set to 3 in the experiments.

This 1D synthetic problem features several aspects relevant to the empirical assessment of EVAE. Firstly,

[2] These problems are publicly available at https://github.com/vberger/compvae .

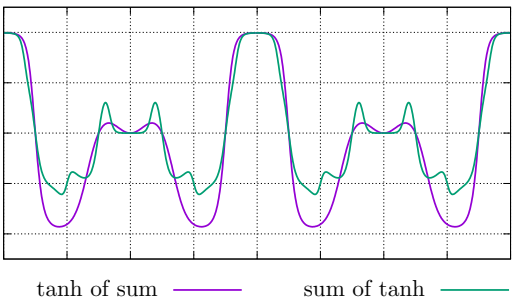


Figure 2: Non-linear part-to-whole aggregation (purple) compared to the sum of non-linear perturbations of the parts (green). Better seen in color. Both curves involve a non-linear transform factor $C = 3$.

the impact of adding or removing one part can be visually assessed as it changes the whole curve: the general magnitude of the whole curve is roughly proportional to its number of parts. Secondly, each part involves, besides its category ℓ_i, some intrinsic variations of its amplitude and phase. Lastly, the whole x is not the sum of its parts (Fig. 2).

The generative model $p_\theta(x|z, \sum_i w_i)$ is defined as a Gaussian distribution $\mathcal{N}(\mu; \Delta(\sigma))$, the vector parameters μ and σ of which are produced by the neural network .

In the 2D synthetic problem, each category in Λ is composed of one out of five colors ($\{red, green, blue, white, black\}$) associated with a location (x, y) in $[0, 1] \times [0, 1]$. Each ℓ_i thus is a colored site, and its internal variability is its intensity. The whole x associated to a set of ℓ_is is an image, where each pixel is colored depending on its distance to the sites and their intensity (Fig. 3). Likewise, the observation model $p_\theta(x|z, \sum_i w_i)$ is a Gaussian distribution $\mathcal{N}(\mu; \Delta(\sigma))$, the parameters μ and σ of which are produced by the neural network. The observation variance is shared for all three channel values (red, green, blue) of any given pixel.

The 2D problem shares with the 1D problem the fact that each part is defined from its category ℓ_i (resp. a frequency, or a color and location) on the one hand, and its specifics on the other hand (resp, its amplitude and frequency, or its intensity); additionally, the whole is made of a set of parts in interaction. However, the 2D problem is significantly more complex than the 1D, as will be discussed in section 5.2.

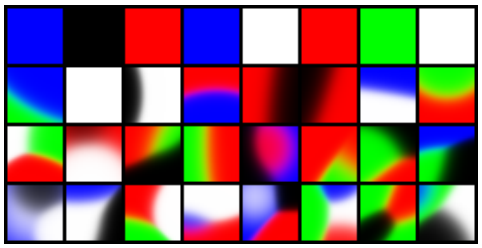


Figure 3: 2D visual synthetic examples, including 1 to 4 sites (top to bottom). Note that when neighbor sites have same color, the image might appear to have been generated with less sites than it actually has.

4.3 Experimental setting

EVAE is trained as a mainstream VAE, except for an additional factor of difficulty: the varying number of latent variables (reflecting the varying number of parts) results in a potentially large number of latent variables. This large size and the model noise in the early training phase can adversely affect the training procedure, and lead it to diverge. The training divergence is prevented using a batch size set to 256. The neural training hyperparameters are dynamically tuned using the Adam optimizer [KB14] with $\alpha = 10^{-4}$, $\beta_1 = 0.5$ and $\beta_2 = 0.9$, which empirically provide a good compromise between training speed, network stability and good convergence. On the top of Adam, the annealing of the learning rate α is achieved, dividing its value by 2 every 20,000 iterations, until it reaches 10^{-6}.

For both problems, the data is generated on the fly during the training, preventing the risk of overfitting. The overall number of iterations (batches) is up to[3] 500,000. The computational time on a GPU GTX1080 is 1 day for the 1D problem, and 2 days for the 2D problem.

Empirically, the training is facilitated by gradually increasing the number K of parts in the datapoints x. Specifically, the number of parts is uniformly sampled in $[[1, K]]$ at each iteration, with $K = 2$ at the initialization and K incremented by 1 every 3,000 iterations, up to 16 parts in the 1D problem and 8 in the 2D problem.

5 EVAE: Empirical validation

This section reports on the proposed proofs of concept of the EVAE approach.

[3] Experimentally, networks most often converge much earlier.

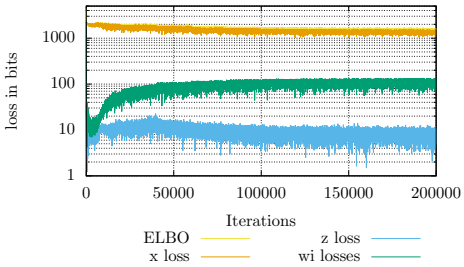


Figure 4: EVAE, 1D problem: Losses of the latent variables respectively associated to the parts (w_i, green), to the whole (z, blue), and the reconstruction loss of x (yellow), in log scale. Better seen in color.

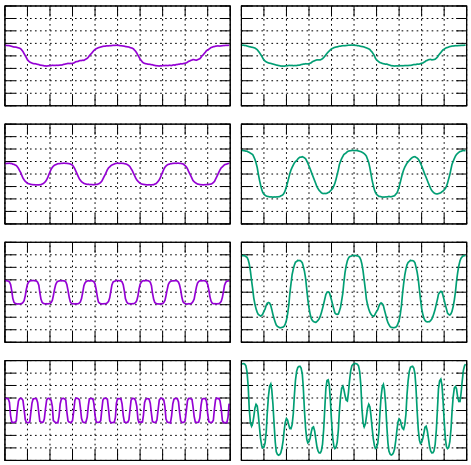


Figure 5: EVAE, 1D problem: Ensemblist recomposition of the whole (right column) from the parts (left column). On each row is given the part (left) and the whole (right) made of this part and all above parts.

5.1 1D Proof of Concept

Fig. 4 displays in log-scale the losses of the w_is and z latent variables along time, together with the reconstruction loss and the overall ELBO loss summing the other three (Eq. (4)). The division of labor between the w_is and the z is seen as the quantity of information stored by the w_is increases to reach a plateau of circa 100 bits, while the quantity of information stored by z steadily decreases to around 10 bits. As conjectured (section 3.1), z is a low-information variable.

Note that the x reconstruction loss remains high, with a high ELBO even at convergence time, although the generated curves "look good". This fact is explained from the high entropy of the data: on the top of the specifics of each part (its amplitude and phase), x is described as a T-length sequence: the temporal discretization of the signal increases the variance of x and thus causes a high entropy, which is itself a lower bound for the ELBO. Note that a large fraction of this entropy is accurately captured by EVAE through the variance of the generative model $p_\theta(x|z, \widetilde{w})$.

The ability of "ensemblist disentanglement" is visually demonstrated on Fig. 5: considering a set of ℓ_i, the individual parts w_i are generated (Fig. 5, left) and gradually integrated to form a whole x (Fig. 5, right) in a coherent manner.

The size-generality property is satisfactorily assessed as the model could be effectively used with a number of parts K ranging up to 30 (as opposed to 16 during the training) without requiring any re-training or other modification of the model (results omitted for brevity).

5.2 2D Proof of Concept

As shown in Fig. 6, the 2D problem is more complex. On the one hand, a 2D part only has a local impact on x (affecting a subset of pixels) while a 1D part has a global impact on the whole x sequence. On the other hand, the number of parts has a global impact on the range of x in the 1D problem, whereas each pixel value ranges in the same interval in the 2D problem. Finally and most importantly, x is of dimension 200 in the 1D problem, compared to dimension $3,072$ ($3 \times 32 \times 32$) in the 2D problem. For these reasons, the latent variables here need to store more information, and the separation between the w_i (converging toward circa 200-300 bits of information) and z (circa 40-60 bits) is less clear.

Likewise, x reconstruction loss remains high, although the generated images "look good", due to the fact that the loss precisely captures the discrepancies in the pixel values that the eye does not perceive.

Finally, the ability of "ensemblist disentanglement" is inspected by incrementally generating the whole x from a set of colored sites (Fig. 7). The top row displays the colors of $\ell_1 \dots \ell_5$ from left to right. On the second row, the i-th square shows the whole x composed of $\ell_1 \dots \ell_i$ (rows 3 to 5 show the variability of the xs constructed from the same parts). While the whole x generally reflects the associated set of parts, some advents of black and white glitches are also observed (for instance on the third column, rows 3 and 5). These glitches are blamed on the saturation of the network (as

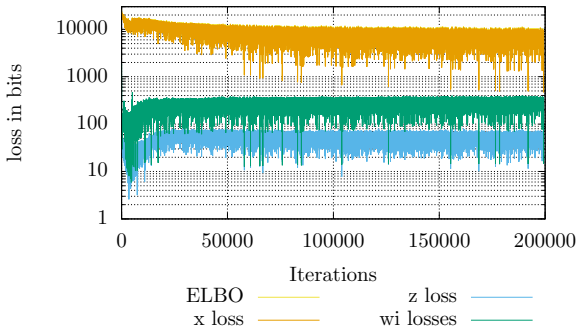


Figure 6: EVAE, 2D problem: Losses of the latent variables respectively associated to the parts (w_i, green), to the whole (z, blue), and the reconstruction loss of x (yellow), in log scale. Better seen in color.

black and white respectively are represented as $(0, 0, 0)$ and $(1, 1, 1)$ in RGB), since non linear combinations of colors are used for a good visual rendering[4].

6 Discussion

The main contribution of the paper is the generative framework EVAE, to our best knowledge the first generative framework able to support the generation of data based on a multi-ensemble $\{\ell_i\}$. Built on the top of the celebrated VAE, EVAE learns to optimize the conditional distribution $p_\theta(x|\{\ell_i\})$ in a theoretically sound way, through introducing latent variables (one for each part ℓ_i), enforcing their order-invariant aggregation and learning another latent variable to model the interaction of the parts. Two proofs of concepts for the approach, respectively concerning a 1D and a 2D problem, have been established with respectively very satisfactory and satisfactory results.

This work opens several perspectives for further research. A first direction in the domain of computer vision consists of combining EVAE with more advanced image generation models such as PixelCNN [vdOKE+16] in a way similar to PixelVAE [GKA+17], in order to generate realistic images involving a predefined set of elements along a consistent layout.

A second perspective is to make one step further toward the training of fully programmable generative models. The idea is to incorporate explicit biases on the top of the distribution learned from unbiased data,

[4]Color blending in the data generation is done taking into account gamma-correction.

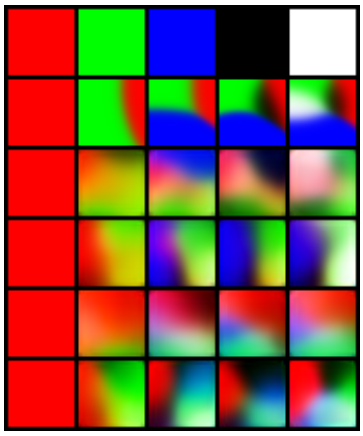


Figure 7: EVAE, 2D problem: Ensemblist recomposition of the whole from parts ℓ_1 to ℓ_5 (1st row). On row 2, the i-th square depicts the whole x defined from ℓ_1 to ℓ_i as generated by the ground truth. Rows 3-6 show different realizations of the same combination by the trained EVAE - see text. Best viewed in colors.

to be able to sample the desired sub-spaces of the data space. In the motivating application domain of electric consumption for instance, one would like to sample the global consumption curves associated with high consumption peaks, that is, to bias the generation process toward the top quantiles of the overall distribution.

Acknowledgments

This work was funded by the ADEME #1782C0034 project *NEXT*.

The authors would like to thank Balthazar Donon and Corentin Tallec for the fruitful exchanges that occurred during the preparation of this work.

References

[AS18] Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[CDH+16] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[CKD+15] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2980–2988. 2015.

[CRLG+18] John D. Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. *International Conference on Machine Learning*, 2018.

[CWBO19] Jan Chorowski, Ron J. Weiss, Samy Bengio, and Aäron van den Oord. Unsupervised speech representation learning using WaveNet autoencoders. URL: *http://arxiv.org/abs/1901.08810*, 2019.

[DBP+17] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *ICLR*, 2017.

[GBR+12] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13:723–773, 2012.

[GKA+17] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. PixelVAE: A latent variable model for natural images. *ICLR*, 2017.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[GSR+17] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272, 2017.

[HMP+17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2017.

[HZG17] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1878–1889. Curran Associates, Inc., 2017.

[JGJS99] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014.

[KRMW14] Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. *NIPS*, 2014.

[KSB17] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *ICLR*, 2017.

[KW13] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2013.

[KWKT15] Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2539–2547. Curran Associates, Inc., 2015.

[LOW+18] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. URL: *http://arxiv.org/abs/1809.02165*, 2018.

[LSL+15] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. URL: *http://arxiv.org/abs/1511.00830*, 2015.

[MGB+18] Daniel Moyer, Shuyang Gao, Rob Brekelmans, Aram Galstyan, and Greg Ver Steeg. Invariant representations without adversarial training. In *Advances in Neural Information Processing Systems*, pages 9084–9093, 2018.

[MO14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. URL: *http://arxiv.org/abs/1411.1784*, 2014.

[MSSW16] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International Conference on Machine Learning*, pages 1445–1453, 2016.

[Pra12] Dilip K Prasad. Survey of the problem of object detection in real images. *International Journal of Image Processing (IJIP)*, 6(6):441, 2012.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*, 2015.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

[SGT+09] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[SRM+16] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *NIPS*, pages 3738–3746, 2016.

[SZT17] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. URL: *http://arxiv.org/abs/1703.00810*, 2017.

[TPB00] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *CoRR*, physics/0004057, 2000.

[TYL17] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning GAN for pose-invariant face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1415–1424, 2017.

[vdOKE+16] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu koray, Oriol Vinyals, and Alex Graves. Conditional image generation with PixelCNN decoders. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4790–4798. Curran Associates, Inc., 2016.

[WGZ+18] Stefan Webb, Adam Golinski, Rob Zinkov, Siddharth Narayanaswamy, Tom Rainforth, Yee Whye Teh, and Frank Wood. Faithful inversion of generative models for effective amortized inference. In *Advances in Neural Information Processing Systems*, pages 3070–3080, 2018.

[ZWS+13] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333, 2013.

A Stochastic Multiple Gradient Descent Algorithm

Quentin Mercier *[1]

[1]Télécom ParisTech

10 avril 2019

Résumé

In this article, we propose a new method for multiobjective optimization problems in which the objective functions are expressed as expectations of random functions. The present method is based on an extension of the classical stochastic gradient descent (SGD) algorithm through the construction of a descent direction common to all specified objective functions. The use of this common descent vector and the Pareto stationarity definition into the stochastic gradient algorithm makes the algorithm able to solve multiobjective problems without any a priori over the objectives. The almost sure convergence of this new algorithm is given considering the classical stochastic gradient algorithm hypothesis. The algorithm efficiency is illustrated on a set of classical benchmarks with diverse complexity and its performance is compared to two deterministic algorithm (*NSGA-II* ,*DMS*) coupled with a Monte Carlo expectation estimator.

Mots-clef : Multiple objective programming, Multiobjective stochastic optimization, Stochastic gradient algorithm, Multi gradient descent algorithm, Common descent vector

1 Introduction

Optimization under uncertainty has known important advances since the second-half of the 20th century [Dan55, BZ70, KKST17] and various approaches have been proposed including robust optimization, which encompasses today a rather large field of robustness concepts such as the "worst case" or the "mean and variance" concepts [GMT14], and stochastic optimization where uncertain parameters are modeled through random variables with a given distribution and where the probabilistic information is directly introduced in the numerical approaches. In that context the uncertain multiobjective problems are written in terms of the expectation of each objective. In our paper we shall focus on this last interpretation of the optimization problem. Considering single objective stochastic optimization problems, a large variety of numerical approaches [Sah04, RHT08] can be found in the literature. Two main distinct approaches exist, one based on stochastic approximations such as the Robbins Monro algorithm and the various stochastic gradient approaches [RM51, Erm83, EW88], the second one based on scenario approaches [Sha03, NS06], the latter being more frequently applied for chance constrained problems.

Regarding stochastic multiobjective optimization the literature is less prolific : the various approaches proposed are based on classical deterministic algorithm such as genetic algorithms coupled with a robust formulation where the random quantities appearing (such as the mean values or standard deviations) are either obtained analytically for simple objectives [FW14, DG05, CCdMMR04] or estimated using a sample averaging approach using scenarios [FX11, BC14, MM05, Löh16, DHX17, WGZW15]. In this paper, we propose a new algorithm for constructing the set of Pareto stationary points of a multiobjective optimization problem written in terms of the mean objective functions. The method is based on the use of the *MGDA* algorithm [Dés12] and more precisely on the existence of a common descent vector analogous to the steepest descent vector, together with a stochastic gradient algorithm. Convergence of this new algorithm will be given as well with several illustrations. The paper is organized as follows. In section 2, the general formalism of multiobjective optimization is given while in the mean time a common descent vector is constructed. In section 3, we introduce the Stochastic Multiple Gradient Descent Algorithm (*SMGDA*). Then we state the convergence property that this new algorithm has. In section 4 illustrations of the *SMGDA* algorithm will

*quentin.mercier@telecom-paristech.fr

be given and compared to the classical Sample Average Approximation (SAA) approach [Sha03].

2 Mulitobjective optimization and common descent vector

Consider m functions $f_j : \mathbb{R}^n \to \mathbb{R}$, $j = 1, ..., m$. Multiobjective optimization consists in solving the following problem without making any a priori over the importance of an objective over an other

$$\min_{\mathbf{x}\in\mathbb{R}^n} \{f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x})\}. \quad (1)$$

In order to solve this problem, one have to change the classical order relation used in mono-objective optimization to a new partial order which is related to the definition below

definition 1. *Considering an optimization problem of form* (1)*, one says that* $\mathbf{x}_1$ *dominates in the sens of Pareto the point* $\mathbf{x}_2$ *if* $\mathbf{x}_1$ *is at least as good as* $\mathbf{x}_2$ *over all the objectives and if* $\mathbf{x}_1$ *is strictly better than* $\mathbf{x}_2$ *for at least one of the objective :*

$$\begin{cases} \forall i \in [\![1, m]\!],\ f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2) \\ \exists \ell \in [\![1, m]\!],\ f_\ell(\mathbf{x}_1) < f_\ell(\mathbf{x}_2) \end{cases} \quad (2)$$

The methodology exposed in this paper relies on the construction of a descent direction that is common to all objectives at the same time.

definition 2. *Let* $\mathbf{d}$ *be a vector of* $\mathbb{R}^n$*, one says that* $\mathbf{d}$ *is a common descent vector to the objectives* $\{f_i\}$ *with* $i \in [\![1, m]\!]$ *in* $\mathbf{x}$ *if the functions* f_i *decrease monotically in the direction* $\mathbf{d}$ *over a segment*

$$\exists \ell_0 \in \mathbb{R}^+,\ \forall \ell \in [0, \ell_0];\ f_i(\mathbf{x}+\ell\mathbf{d}) \leq f_i(\mathbf{x}),\ \forall i \in [\![1, m]\!].$$

From now on we consider that the objective functions f_j are differentiable and their gradients ∇f_j exist in any point of $\mathbb{R}^n$. We now introduce the convex hull of the union of the gradients calculated at a design point $\mathbf{x}$.

$$\mathcal{C}(\mathbf{x}) = Conv\left(\bigcup_{j=1}^{m} \nabla f_j(\mathbf{x})\right) = \left\{ \begin{array}{c} \mathbf{u} \in \mathbb{R}^n, \mathbf{u} = \sum_{j=1}^m \alpha_j \nabla f_j(\mathbf{x}), \\ \\ \alpha_j \geq 0,\ \sum_{j=1}^m \alpha_j = 1 \end{array} \right\}$$

We now introduce the concept of Pareto stationarity which will be used extensively in the next sections.

definition 3. *We say that a point* $\mathbf{x}^\star$ *in the design space is a Pareto stationary point if the nul vector is in the convex hull of the gradients*

$$\mathbf{0} \in \mathcal{C}(\mathbf{x}^\star).$$

The algorithm proposed in this paper is based on the construction of a common descent vector at each step which is used as a descent direction.

Proposition 1. *[MPD18b] Let* $\mathbf{x}$ *be a point of the design space, the minimum norm element* $\xi(\mathbf{x})$ *of* $\mathcal{C}(\mathbf{x})$ *is a common descent vector to the objective functions* $f_j(\mathbf{x})$

$$\xi(\mathbf{x}) = \text{argmin}\,\{\|\mathbf{u}\|,\ \mathbf{u} \in \mathcal{C}(\mathbf{x})\}.$$

Remark 1. *The proposition can be extended to* f^0*-pseudo-convex functions [MKE10] by replacing the union of gradients with the union of Clarke subdifferentials in the definition of* $\mathcal{C}$*. Convergence property given later in the paper [MPD18a] remains unchanged.*

3 Stochastic Multiple Gradient Descent Algorithm (*SMGDA*)

Let $(\Omega, \mathcal{A}, \mathbb{P})$ be an abstract probabilistic space, and $W : \Omega \to \mathbb{R}^d$, $\omega \mapsto W(\omega)$ a given random vector. We denote μ the distribution of the random variable W and $\mathcal{W}$ its image space $W(\Omega) \subset \mathbb{R}^d$. Let $W_1, ..., W_p, ...$ be independent copies of the random variable W which will be used to generate independent random samples with distribution μ.

Throughout the paper the standard inner product on $\mathbb{R}^n$ will be used and denoted $\langle \cdot, \cdot \rangle$, the norm being denoted $\| \cdot \|$.

Consider m functions $f_j : \mathbb{R}^n \times \mathcal{W} \to \mathbb{R}$, $j = 1, ..., m$. The problem addressed in this paper is to solve the mean multiobjective optimization problem written

$$\min_{\mathbf{x}\in\mathbb{R}^n} \{\mathbb{E}[f_1(\mathbf{x}, W), \mathbb{E}[f_2(\mathbf{x}, W)], ..., \mathbb{E}[f_m(\mathbf{x}, W)]\}. \quad (3)$$

More precisely we want to construct a set of points that belongs to the associated Pareto set. As it is written, problem (3) is a deterministic problem but in general the objective function expectations are not known. A classical approach is to replace each expectancy by an estimator built using independent samples w_k of the random variable W [BC14, FX11]. As for stochastic gradient algorithms, the algorithm we propose does not need to calculate the mean objective functions and is

only based on the values of the stochastic functions gradients. The classical stochastic gradient algorithm is based on a descent direction given by the objective function gradient. Here the descent vector which will be used is the common descent vector constructed in the previous section. In the stochastic context this common descent vector is random and defined by the random convex combination

$$\xi(\mathbf{x}, W) = \sum_{j=1}^{m} \alpha_j(\mathbf{x}, W) \nabla f_j(\mathbf{x}, W) \ a.s., \ \mathbf{x} \in \mathbb{R}^n \quad (4)$$

with

$$\sum_j \alpha_j(\mathbf{x}, W) = 1 \ a.s. \quad (5)$$

by construction.

The flow chart of *SMGDA* (*Stochastic Multi-Gradient Descent Algorithm*) algorithm is described below.

Algorithm: *SMGDA*

input:
- An initial point $\mathbf{X}_0$ of the design space
- A number of iterations N
- A σ-sequence $\{\epsilon_k\}_{k\in\mathbb{N}}$

begin
- $\mathbf{X} = \mathbf{X}_0$;
- **for** $k \in [\![1, N]\!]$ **do**
  - Generate a sample w_k of random variable W_k;
  - Evaluate the objective functions and their gradients $(\mathbf{X}_{k-1}, w_k) \longrightarrow (f_j(\mathbf{X}_{k-1}, w_k), \nabla f_j(\mathbf{X}_{k-1}, w_k))$;
  - Calculate the common descent vector $\xi(\mathbf{X}_{k-1}, w_k)$;
  - Update the current parameter values : $\mathbf{X}_k = \mathbf{X}_{k-1} - \epsilon_k \xi(\mathbf{X}_{k-1}, w_k)$.

The notation $\mathcal{P}_D^\star$ (resp. $\mathcal{P}_O^\star$) will denote the Pareto solution set (resp. the Pareto front). For any $\mathbf{x} \in \mathbb{R}^n$ the notation $\mathbf{x}^\perp$ will denote an element of the Pareto set which minimizes the distance between the point $\mathbf{x}$ and a point of the Pareto set $\mathcal{P}_D{}^\star$

$$\mathbf{x}^\perp \in \underset{\mathbf{u}\in\mathcal{P}_D^\star}{\operatorname{argmin}} \{\|\mathbf{x} - \mathbf{u}\|\}. \quad (6)$$

The convergence result relies upon the following hypotheses.

1. Problem (3) admits a nonempty Pareto solution set $\mathcal{P}_D^\star$.
2. The random variables $f_j(\mathbf{x}, W)$ are integrable for $j = 1, ..., m$ and for all $\mathbf{x} \in \mathbb{R}^n$.
3. The functions $\mathbf{x} \mapsto f_j(\mathbf{x}, W) : \mathbb{R}^n \to \mathbb{R}$ are convex and their derivatives exist almost surely for $j = 1, ..., m$.
4. The partial gradient of function f_j with respect to $\mathbf{x}$ is almost surely uniformly bounded by a strictly positive real number M_j
$$\|\nabla f_j(\mathbf{x}, W)\| \leq M_j \ a.s., \ \mathbf{x} \in \mathbb{R}^n.$$
5. For any objective function f_j, there exists a positive real number c_j such that for any $\mathbf{x}$ in $\mathbb{R}^n$ the following relation holds
$$f_j(\mathbf{x}, W) - f_j(\mathbf{x}^\perp, W) \geq \frac{c_j}{2}\|\mathbf{x} - \mathbf{x}^\perp\|^2 \, a.s. \ ;$$
j=1,. . .,m.
6. The sequence $\{\epsilon_k\}_{k\in\mathbb{N}}$ is a σ-sequence
$$\sum_{k=0}^{\infty} \epsilon_k = \infty$$
$$\sum_{k=0}^{\infty} \epsilon_k^2 < \infty.$$

Theorem 1. *[MPD18b] The sequence of random variables $\mathbf{X}_0, \mathbf{X}_1, ..., \mathbf{X}_n$ constructed using the SMGDA algorithm converges almost surely towards a point $\mathbf{X}^\perp$ of the Pareto set $\mathcal{P}_D^\star$*

$$\mathbb{P}\left(\left\{\lim_{k\to\infty} (\mathbf{X}_k - \mathbf{X}_k^\perp) = 0\right\}\right) = 1.$$

4 Numerical Experiments

The efficiency and the reliability of the method is assessed by comparing the solution obtained by *SMGDA* and by two other solvers : *NSGA-II* [DPAM02], and *DMS* [CMVV11] on several classical deterministic benchmark problems described in [HHBW06]. The problems are chosen in order to present different situations : convex, nonconvex and discontinuous Pareto sets. Uncertainties are added to each problem by introducing random variables into the objective functions. Since the expectations appearing in the optimization problem described by equation (3)

are not available, a sample average approximation approach is used for *NSGA-II* and *DMS* in order to evaluate them :

$$\mathbb{E}[f(\mathbf{x}, W)] \approx \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}, w_i) \tag{7}$$

where w_i are independent samples of the random variable W. The number N of samples plays a crucial role in the efficiency of the algorithm : a too small value results in a wide confidence interval and a poor estimate of the objective function, while an excessive value results in a dramatic increase of the computational cost. The test cases are conducted taking into account a budget based on the maximum number of calls to the objective functions. In order to compare the performance of the three algorithms tested in this section, two classical indicators are introduced : a performance indicator called Purity, which compares the number of non-dominated points an algorithm is able to find to a reference front built using the results of the three optimizers, and the well known Hypervolume indicator [ABBZ09] which gives an indication on both the spreading of the front and its quality by calculating the sum of the hypervolumes generated between all non-dominated points and a reference point taken in the objective space. For both indicators the higher the score is the better the performance is.
The tuning of *NSGA-II* and *DMS* parameters (including the number of samples used for the sample average approximation) is not straightforward and we used an auxiliary genetic algorithm in order to find the parameter values which maximizes the resulting Hypervolume measure for each problem. Due to the stochastic nature of the problems the fitness considered is the mean value of the resulting Hypervolume measure estimated on a sample of 5 independent runs for the same set of parameter values. The tuning optimizer is run for a population of 20 individuals and 15 generations.
The results of the test cases are analyzed using performance profiles which gives an indication of performance on the overall set of problems [DM02].

The three algorithms are now compared using several benchmark tests described in in table 1. The last two columns give the number of total calls budget during the optimization process for each algorithm tested. The number of calls allowed for *SMGDA* is set to be 10 times less than for *DMS* and *NSGA-II* .

In this section the pareto front are not drawn for each problem, but performance profiles using Purity and Hypervolume metrics are used in order to compare the performance of the three algorithms. The performance profiles correspond to a cumulative distribution function that gives an indication of the percentage of problems considered solved for a certain threshold τ of the ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,\bar{s}},\ \bar{s} \in \mathcal{S}\}}$$

where p is an optimization problem belonging the set $\mathcal{P}$ of benchmark problems addressed, s is the solver used in $\mathcal{S} = \{SMGDA\ , DMS\ , NSGA\text{-}II\ \}$ and t is a performance indicator for which the lower score is the better the performance is. Thus, in this section, it is actually the inverse of Purity and Hypervolume metrics which are used. This leads, for each solver, to the following expression of the cumulative distribution function $\rho(\tau)$

$$\rho_s(\tau) = \frac{1}{Card\,(\mathcal{P})} \times Card\,(\{p \in \mathcal{P},\ r_{p,s} \leq \tau\})\,.$$

Thus, for each solver s, the value of $\rho_s(1)$ is the number of problems for which the performance of algorithm s is superior to the other two.

For τ equal to 1, *SMGDA* outperforms the two other algorithms. Regarding the performance profiles based on the Purity metric represented on Figure 1, *SMGDA* has a better performance for eight test problems and is the only algorithm able to reach $\rho_s = 1$. Which means that *NSGA-II* and *DMS* have not been able generate non-dominated points on some of the addressed problems for the allocated budget. Because

Table 1 – Benchmark problems extracted from [HHBW06] [ZDT00] [JOS01]

Nom	$\dim(\mathbf{x})$	$\dim(W(\omega))$	Objective calls
MOP2	15	30	10^5
MOP3	2	18	10^4
MOP6	2	3	10^4
ZDT{1,2,3}	32	30	$5 \cdot 10^5$
JOS1	30	60	$5 \cdot 10^5$
JOS2	30	32	$5 \cdot 10^5$
SCH1	1	4	10^4
IM1	2	3	10^4

SMGDA may converge towards Pareto stationary but not optimal points, *SMGDA* obtains a good score using the Purity metric for all the problems except for the two multimodal test cases. But the algorithm reach nevertheless the value of $\rho_s = 1$ for $\tau = 2$ which means that *SMGDA* has been worst regarding the Purity metric than the best performing algorithm on those particular test cases by a factor of only 2. Whereas, for certain test cases, *DMS* (resp. *NSGA-II*) can result in a Purity score up to 10 (resp. 100) times lower than the winning algorithm. This demonstrates the capability of *SMGDA* to perform well and to give good quality results for all the benchmark problems addressed.

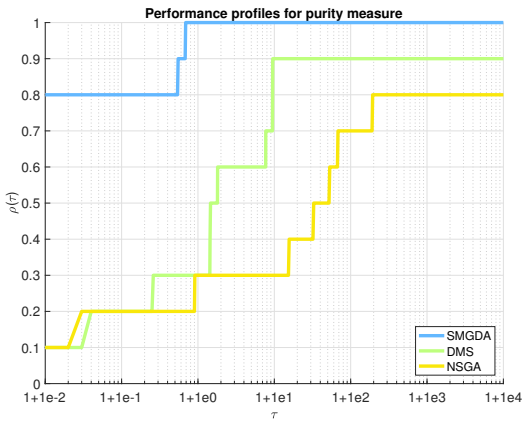


FIGURE 1 – Performance profile of the Purity indicator

Since the parameters of both *DMS* and *NSGA-II* have been optimized specifically for the Hypervolume metric, the two algorithms show a better performance profile than the ones constructed for the Purity metric as it is illustrated in Figure 2. *NSGA-II* algorithm, especially, outperforms the other two algorithms for three test problems. Even if *SMGDA* performance is lower for the Hypervolume metric, it is nevertheless rather efficient since the performance reaches $\rho_s = 1$ for a very low value of τ. For its less performing test case, *SMGDA* performance score was only 1.031 times less than the best algorithm performance.

The performance profiles presented in this section show that *SMGDA* can compete successfully with classical algorithms used for multiobjective optimization problems, at least for the two performance metric introduced. The numerical efficiency of *SMGDA* comes mainly from the fact that no estimator construction is necessary to evaluate the objective functions. Moreover the algorithm efficiency does not depend on the number of random variables introduced in the objective functions nor on the number of objective functions. The weakness of the method is the necessity to have the gradient analytic expressions, their numerical calculation would of course increase the computation time.

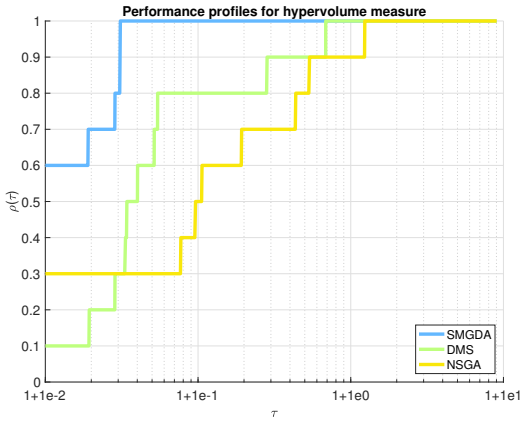


FIGURE 2 – Performance profile of the Hypervolume indicator

5 Conclusions

In this article we have proposed a novel algorithm for solving a stochastic multiobjective optimization problem. It is based on two ingredients : a common random descent vector and an extension of the classical stochastic gradient algorithm. Because the algorithm necessitates only a single iteration loop, it is less time demanding than classical approaches based on sample averaging approximation methods. The almost sure convergence has been proved based on rather restrictive assumptions. As it is the case for stochastic gradient algorithms, no efficient stopping criterion exists. Comparisons with *NSGA-II* and *DMS* on a set of benchmark problems have shown the very good behaviour of the proposed method which needs much less iterations to converge than the two other solvers tested. Compared to a genetic algorithm there is no exchange of information between the initial points, which may a priori seem to yield a suboptimal decision but which renders the algorithm entirely and readily parallelizable.

Références

[ABBZ09] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Theory of hypervolume indicator : Optimal μ-distributions and the choice of the reference point. *Proceedings of the Tenth ACM SIGEVO Workshop Foundations of Genetic Algorithms*, pages 87–102, 2009.

[BC14] H. Bonnel and J. Collonge. Stochastic optimization over a pareto set associated with stochastic multiobjective optimization problem. *Journal of Optimization Theory and Applications*, 162(2) :405–427, 2014.

[BZ70] R. Bellman and L. Zadeh. Decision-making in fuzzy environment. *Management Science*, 17 :141–161, 1970.

[CCdMMR04] R. Caballero, E. Cerdá, del Mar Muñoz, and L. Rey. Stochastic approach versus multiobjective approach for obtaining efficient solutions in stochastic multiojective programming problems. *European Journal of Operational Research*, 158 :633–648, 2004.

[CMVV11] A. L. Custódio, F. A. Madeaira, A. I. F. Vaz, and L. N. Vincente. Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization*, 21(3) :1109–1140, 2011.

[Dan55] B. Dantzig. Linear programming under uncertainty. *Management Science*, 1 :197–206, 1955.

[Dés12] Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5–6) :313 – 318, 2012.

[DG05] Kalyanmoy Deb and Himanshu Gupta. Searching for robust pareto-optimal solutions in multi-objective optimization. In CarlosA. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 150–164. Springer Berlin Heidelberg, 2005.

[DHX17] J. E. Diaz, J. Handl, and D.-L. Xu. Evolutionary robust optimization in production planning interactions between number of objectives, sample size and choice of robustness measure. *Computers & Operations Research*, 79 :266–278, 2017.

[DM02] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2) :201–213, 2002.

[DPAM02] Kalyanmoy Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197, 2002.

[Erm83] Y. Ermoliev. Stochastic quasigradient methods and their application to systems optimization. *Stochastics*, 9 :1–36, 1983.

[EW88] Y. Ermoliev and R. Wets. *Numerical Technics for Stochastic Optimization.* Springer Verlag, 1988.

[FW14] J. Fliege and R. Werner. Robust multiobjective optimization & applications in portfolio optimization. *European Journal of Operational Research*, 234 :422–433, 2014.

[FX11] J. Fliege and H. Xu. Stochastic multiobjective optimization : Sample average approximations and applications. *Journal of Optimization Theory and Applications*, 151(1) :135–162, 2011.

[GMT14] V. Gabrel, C. Murat, and A. Thiele. Recent advances in robust optimization : An overview. *European Journal of Operational Research*, 235 :471–483, 2014.

[HHBW06] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10 :477–506, 2006.

[JOS01] Y. Jin, M. Olhofer, and B. Sendhoff. Dynamic weighted aggregation for evolutionary multiobjective optimization : Why does it work and how ? *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1042–1049, 2001.

[KKST17] K. Klamroth, E. Kbis, A. Schbel, and C. Tammer. A unified approach to uncertain optimization. *European Journal of Operational Research*, 260 :805–819, 2017.

[Löh16] N. Löhndorf. An empirical analysis of scenario generation methods for stochastic optimization. *European Journal of Operational Research*, 255 :121–132, 2016.

[MKE10] M. M. Mäkelä, N. Karmitsa, and V.-P. Eronen. On generalized pseudo and quasi-convexities for nonsmooth functions. Technical Report No 989, Turku Center for Computer Science, 2010.

[MM05] C. A. Mattson and A. Messac. Pareto frontier based concept selection under uncertainty, with visualization. *Optimization and Engineering*, 6 :85–115, 2005.

[MPD18a] Q. Mercier, F. Poirion, and Jean-Antoine Désidéri. Nonconvex multiobjective design optimization under uncertainty : a descent algorithm. application to sandwich plate design and reliability. *Engineering Optimization*, 2018.

[MPD18b] Q. Mercier, F. Poirion, and Jean-Antoine Désidéri. A stochastic multiple gradient descent algorithm. *European Journal of Operational Research*, 271(3) :808–817, 2018.

[NS06] A. Nemirovski and A. Shapiro. Scenario approximation of chance constraints. In *Probabilistic and Randomized Methods for Design under Uncertainty*. Springer, 2006.

[RHT08] R. Roy, S. Hinduja, and R. Teti. Recent advances in engineering design optimization : challenge and future trends. *Manufactering Technology*, 57 :697–715, 2008.

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[Sah04] Nikolaos V. Sahinidis. Optimization under uncertainty : state-of-the-art and opportunities. *Computers and Chemical Engineering*, 28(6–7) :971 – 983, 2004. {FOCAPO} 2003 Special issue.

[Sha03] Alexander Shapiro. Monte carlo sampling methods. In *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*, pages 353 – 425. Elsevier, 2003.

[WGZW15] Z. Wang, J. Guoa, M Zheng, and Y. Wang. Uncertain multiobjective traveling salesman problem. *European Journal of Operational Research*, 241 :478–489, 2015.

[ZDT00] E. Zitzler, Kalyanmoy Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms : Empirical results. *Evolutionary Computation*, 8 :173–195, 2000.

Infinite Task Learning in RKHSs

Romain Brault[1], Alex Lambert[2], Zoltán Szabó[3], Maxime Sangnier[4], et Florence d'Alché-Buc[5]

[1]Thales
[2]Télécom ParisTech
[3]UPMC
[4]École Polytechnique
[5]Télécom ParisTech

April 10, 2019

Abstract

Machine learning has witnessed tremendous success in solving tasks depending on a single hyperparameter. When considering simultaneously a finite number of tasks, multi-task learning enables one to account for the similarities of the tasks via appropriate regularizers. A step further consists of learning a continuum of tasks for various loss functions. A promising approach, called *Parametric Task Learning*, has paved the way in the continuum setting for affine models and piecewise-linear loss functions. In this work, we introduce a novel approach called *Infinite Task Learning*: its goal is to learn a function whose output is a function over the hyperparameter space. We leverage tools from operator-valued kernels and the associated Vector-Valued Reproducing Kernel Hilbert Space that provide an explicit control over the role of the hyperparameters, and also allows us to consider new type of constraints. We provide generalization guarantees to the suggested scheme and illustrate its efficiency in cost-sensitive classification, quantile regression and density level set estimation.

Mots-clef: méthodes à noyaux, multi-task learning.

1 Introduction

Several fundamental problems in machine learning and statistics can be phrased as the minimization of a loss function described by a hyperparameter. The hyperparameter might capture numerous aspects of the problem: (i) the tolerance w. r. t. outliers as the ϵ-insensitivity in Support Vector Regression [VGS97], (ii) importance of smoothness or sparsity such as the weight of the $\mathfrak{l}_2$-norm in Tikhonov regularization [TA77], $\mathfrak{l}_1$-norm in LASSO [Tib96], or more general structured-sparsity inducing norms [Bac+12], (iii) Density Level-Set Estimation (DLSE), see for example one-class support vector machines One-Class Support Vector Machine (OCSVM, [Sch+00]), (iv) confidence as examplified by Quantile Regression (QR, [KB78]), or (v) importance of different decisions as implemented by Cost-Sensitive Classification (CSC, [ZE01]).

In various cases including QR, CSC or DLSE, one is interested in solving the parameterized task for several hyperparameter values. Multi-Task Learning [EP04] provides a principled way of benefiting from the relationship between similar tasks while preserving local properties of the algorithms: ν-property in DLSE [GLM13] or quantile property in QR [Tak+06].

A natural extension from the traditional multi-task setting is to provide a prediction tool being able to deal with *any* value of the hyperparameter. In their seminal work, [Tak+13] extended multi-task learning by considering an infinite number of parametrized tasks in a framework called Parametric Task Learning (PTL). Assuming that the loss is piecewise affine in the hyperparameter, the authors are able to get the whole solution path through parametric programming, relying on techniques developed by Hastie et al. [Has+04].

In this paper, we relax the affine model assumption on the tasks as well as the piecewise-linear assumption on the loss, and take a different angle. We propose Infinite Task Learning (ITL) within the framework of function-valued function learning to handle a continuum number of parameterized tasks. For that purpose we leverage tools from operator-valued kernels and the associated Vector-Valued Reproducing Kernel Hilbert Space (vv-RKHS, [Ped57]). The idea is that the output is a

function on the hyperparameters —modelled as scalar-valued Reproducing Kernel Hilbert Space (RKHS)—, which provides an explicit control over the role of the hyperparameters, and also enables us to consider new type of constraints. In the studied framework each task is described by a (scalar-valued) RKHS over the input space which is capable of dealing with nonlinearities. The resulting ITL formulation relying on vv-RKHS specifically encompasses existing multi-task approaches including joint quantile regression [SFd16] or multi-task variants of density level set estimation [GLM13] by encoding a continuum of tasks.

Our **contributions** can be summarized as follows:

- We propose ITL, a novel vv-RKHS-based scheme to learn a continuum of tasks parametrized by a hyperparameter and design new regularizers.
- We prove excess risk bounds on ITL and illustrate its efficiency in quantile regression, cost-sensitive classification, and density level set estimation.

The paper is structured as follows. The ITL problem is defined in Section 2. In Section 3 we detail how the resulting learning problem can be tackled in vv-RKHSs. Excess risk bounds is the focus of Section 4. Numerical results are presented in Section 5. Conclusions are drawn in Section 6.

2 From parameterized to infinite task learning

After introducing a few notations, we gradually define our goal by moving from single parameterized tasks (Section 2.1) to ITL (Section 2.3) through multi-task learning (Section 2.2).

Notations: $\mathbb{1}_S$ is the indicator function of set S. We use the $\sum_{i,j=1}^{n,m}$ shorthand for $\sum_{i=1}^{n}\sum_{j=1}^{m}$. $|x|_+ = \max(x,0)$ denotes positive part. $\mathcal{F}(\mathcal{X};\mathcal{Y})$ stands for the set of $\mathcal{X}\to\mathcal{Y}$ functions. Let $\mathcal{Z}$ be Hilbert space and $\mathcal{L}(\mathcal{Z})$ be the space of $\mathcal{Z}\to\mathcal{Z}$ bounded linear operators. Let $K:\mathcal{X}\times\mathcal{X}\to\mathcal{L}(\mathcal{Z})$ be an operator-valued kernel, i.e. $\sum_{i,j=1}^{n}\langle z_i, K(x_i,x_j)z_j\rangle_{\mathcal{Z}} \geqslant 0$ for all $n\in\mathbb{N}^*$ and $x_1,\dots,x_n\in\mathcal{X}$ and $z_1,\dots,z_n\in\mathcal{Z}$ and $K(x,z)=K(z,x)^*$ for all $x,\ z\in\mathcal{X}$. K gives rise to the Vector-Valued Reproducing Kernel Hilbert Space $\mathcal{H}_K = \overline{\text{span}}\,\{K(\cdot,x)z \mid x\in\mathcal{X},\ z\in\mathcal{Z}\}\subset\mathcal{F}(\mathcal{X};\mathcal{Z})$, where $\overline{\text{span}}\,\{\cdot\}$ denotes the closure of the linear span of its argument. For futher details on vv-RKHS the reader is referred to [Car+10].

2.1 Learning Parameterized Tasks

A *supervised parametrized task* is defined as follows. Let $(X,Y)\in\mathcal{X}\times\mathcal{Y}$ be a random variable with joint distribution $\mathbf{P}_{X,Y}$ which is assumed to be fixed but unknown; we also assume that $\mathcal{Y}\subset\mathbb{R}$. We have access to n independent identically distributed observations called training samples: $\mathcal{S} := ((x_i,y_i))_{i=1}^n \sim \mathbf{P}_{X,Y}^{\otimes n}$. Let Θ be the domain of hyperparameters, and $v_\theta\colon \mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$ be a loss function associated to $\theta\in\Theta$. Let $\mathcal{H}\subset\mathcal{F}(\mathcal{X};\mathcal{Y})$ denote our hypothesis class; throughout the paper $\mathcal{H}$ is assumed to be a Hilbert space with inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$. For a given θ, the goal is to estimate the minimizer of the expected risk

$$R^\theta(h) := \mathbf{E}_{X,Y}[v_\theta(Y,h(X))] \tag{1}$$

over $\mathcal{H}$, using the training sample $\mathcal{S}$. This task can be addressed by solving the regularized empirical risk minimization problem

$$\min_{h\in\mathcal{H}} R_{\mathcal{S}}^\theta(h) + \Omega(h), \tag{2}$$

where $R_{\mathcal{S}}^\theta(h) := \frac{1}{n}\sum_{i=1}^n v_\theta(y_i,h(x_i))$ is the empirical risk and $\Omega:\mathcal{H}\to\mathbb{R}$ is a regularizer. Below we give three examples.

Quantile Regression: In this setting $\theta\in(0,1)$. For a given hyperparameter θ, in Quantile Regression the goal is to predict the θ-quantile of the real-valued output conditional distribution $\mathbf{P}_{Y|X}$. The task can be tackled using the pinball loss [KB78] defined in Eq. (3).

$$\begin{aligned} v_\theta(y,h(x)) &= |\theta - \mathbb{1}_{\mathbb{R}_-}(y-h(x))||y-h(x)|, \\ \Omega(h) &= \tfrac{\lambda}{2}\|h\|_{\mathcal{H}}^2, \quad \lambda>0. \end{aligned} \tag{3}$$

Cost-Sensitive Classification: Our next example considers binary classification ($\mathcal{Y}=\{-1,1\}$) where a (possibly) different cost is associated with each class, as it is often the case in medical diagnosis. The sign of $h\in\mathcal{H}$ yields the estimated class and in cost-sensitive classification one takes

$$\begin{aligned} v_\theta(y,h(x)) &= \left|\tfrac{1}{2}(\theta+1) - \mathbb{1}_{\{-1\}}(y)\right||1-yh(x)|_+, \\ \Omega(h) &= \tfrac{\lambda}{2}\|h\|_{\mathcal{H}}^2, \quad \lambda>0. \end{aligned} \tag{4}$$

The $\theta\in[-1,1]$ hyperparameter captures the trade-off between the importance of correctly classifying the samples having -1 and $+1$ labels. When θ is close to -1, the obtained h focuses on classifying well class -1, and vice-versa. Typically, it is desirable for a physician to choose *a posteriori* the value of the hyperparameter at which he wants to predict. Since this cost can rarely be considered to be fixed, this motivates to learn one model giving access to all hyperparameter values.

Density Level-Set Estimation: Examples of parameterized tasks can also be found in the unsupervised setting. For instance in outlier detection, the goal is to separate outliers from inliers. A classical technique to tackle this task is OCSVM [Sch+00]. OCSVM has a free parameter $\theta \in (0, 1]$, which can be proven to be an upper bound on the fraction of outliers. When using a Gaussian kernel with a bandwidth tending towards zero, OCSVM consistently estimates density level sets [VV06]. This unsupervised learning problem can be empirically described by the minimization of a regularized empirical risk $R_S^\theta(h, t) + \Omega(h)$, solved *jointly* over $h \in \mathcal{H}$ *and* $t \in \mathbb{R}$ with

$$\nu_\theta(t, h(x)) = -t + \frac{1}{\theta}|t - h(x)|_+, \quad \Omega(h) = \tfrac{1}{2}\|h\|^2_{\mathcal{H}}.$$

2.2 Solving a Finite Number of Tasks as Multi-Task Learning

In all the aforementioned problems, one is rarely interested in the choice of a single hyperparameter value (θ) and associated risk $\left(R_S^\theta\right)$, but rather in the joint solution of multiple tasks. The naive approach of solving the different tasks independently can easily lead to inconsistencies. A principled way of solving many parameterized tasks has been cast as a MTL problem [EMP05] which takes into account the similarities between tasks and helps providing consistent solutions. For example it is possible to encode the similarities of the different tasks in MTL through an explicit constraint function [Cil+17]. In the current work, the similarity between tasks is designed in an implicit way through the use of a kernel on the hyperparameters. Moreover, in contrast to MTL, in our case the input space and the training samples are the same for each task; a task is specified by a value of the hyperparameter. This setting is sometimes refered to as multi-output learning [ÁRL12].

Formally, assume that we have p tasks described by parameters $(\theta_j)_{j=1}^p$. The idea of multi-task learning is to minimize the sum of the local loss functions $R_S^{\theta_j}$, i. e.

$$\arg\min_h \sum\nolimits_{j=1}^p R_S^{\theta_j}(h_j) + \Omega(h),$$

where the individual tasks are modelled by the real-valued h_j functions, the overall $\mathbb{R}^p$-valued model is the vector-valued function $h\colon x \mapsto (h_1(x), \ldots, h_p(x))$, and Ω is a regularization term encoding similarities between tasks.

It is instructive to consider two concrete examples:

- In joint quantile regression one can use the regularizer to encourage that the predicted conditional quantile estimates for two similar quantile values are similar. This idea forms the basis of the approach proposed by Sangnier et al. [SFd16] who formulates the joint quantile regression problem in a vector-valued Reproducing Kernel Hilbert Space with an appropriate decomposable kernel that encodes the links between the tasks. The obtained solution shows less quantile curve crossings compared to estimators not exploiting the dependencies of the tasks as well as an improved accuracy.
- A multi-task version of DLSE has recently been presented by Glazer et al. [GLM13] with the goal of obtaining nested density level sets as θ grows. Similarly to joint quantile regression, it is crucial to take into account the similarities of the tasks in the joint model to efficiently solve this problem.

2.3 Towards Infinite Task Learning

In the following, we propose a novel framework called Infinite Task Learning in which we learn a function-valued function $h \in \mathcal{F}(\mathcal{X}; \mathcal{F}(\Theta; \mathcal{Y}))$. Our goal is to be able to handle new tasks after the learning phase and thus, not to be limited to given predefined values of the hyperparameter. Regarding this goal, our framework generalizes the Parametric Task Learning approach introduced by Takeuchi et al. [Tak+13], by allowing a wider class of models and relaxing the hypothesis of piece-wise linearity of the loss function. Moreover a nice byproduct of this vv-RKHS based approach is that one can benefit from the functional point of view, design new regularizers and impose various constraints on the whole continuum of tasks, e. g.,

- The continuity of the $\theta \mapsto h(x)(\theta)$ function is a natural desirable property: for a given input x, the predictions on similar tasks should also be similar.
- Another example is to impose a shape constraint in QR: the conditional quantile should be increasing w. r. t. the hyperparameter θ. This requirement can be imposed through the functional view of the problem.
- In DLSE, to get nested level sets, one would want that for all $x \in \mathcal{X}$, the decision function $\theta \mapsto \mathbb{1}_{\mathbb{R}_+}(h(x)(\theta) - t(\theta))$ changes its sign only once.

To keep the presentation simple, in the sequel we are going to focus on ITL in the supervised setting; unsupervised tasks can be handled similarly.

Assume that h belongs to some space $\mathcal{H} \subseteq \mathcal{F}(\mathcal{X}; \mathcal{F}(\Theta; \mathcal{Y}))$ and introduce an integrated loss func-

tion

$$V(y, h(x)) := \int_{\Theta} \nu(\theta, y, h(x)(\theta)) d\mu(\theta), \tag{5}$$

where the local loss $\nu\colon \Theta \times \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ denotes ν_θ seen as a function of three variables including the hyperparameter and μ is a probability measure on Θ which encodes the importance of the prediction at different hyperparameter values. Without prior information and for compact Θ, one may consider μ to be uniform. The true risk reads then

$$R(h) := \mathbf{E}_{X,Y}\left[V(Y, h(X))\right]. \tag{6}$$

Intuitively, minimizing the expectation of the integral over θ in a rich enough space corresponds to searching for a pointwise minimizer $x \mapsto h^*(x)(\theta)$ of the parametrized tasks introduced in Eq. (1) with, for instance, the implicit space constraint that $\theta \mapsto h^*(x)(\theta)$ is a continuous function for each input x. We show in Proposition 2.1 that this is precisely the case in QR:

Proposition 2.1. *Let X, Y be two random variables respectively taking values in $\mathcal{X}$ and $\mathbb{R}$, and $q\colon \mathcal{X} \to \mathcal{F}([0,1], \mathbb{R})$ the associated conditional quantile function. Let μ be a positive measure on $[0,1]$ such that $\int_0^1 \mathbf{E}\left[\nu_\theta\left(Y, q(X)(\theta)\right)\right] d\mu(\theta) < \infty$. Then $\forall h \in \mathcal{F}\left(\mathcal{X}; \mathcal{F}\left([0,\ 1]\,; \mathbb{R}\right)\right)$*

$$R(h) - R(q) \geqslant 0,$$

where R is the risk defined in Eq. (6).

. Interestingly, the empirical counterpart of the true risk minimization can now be considered with a much richer family of penalty terms:

$$\min_{h \in \mathcal{H}} R_S(h) + \Omega(h), \quad R_S(h) := \frac{1}{n} \sum_{i=1}^{n} V(y_i, h(x_i)). \tag{7}$$

Here, $\Omega(h)$ can be a weighted sum of various penalties

- imposed directly on $(\theta, x) \mapsto h(x)(\theta)$, or
- integrated constraints on either $\theta \mapsto h(x)(\theta)$ or $x \mapsto h(x)(\theta)$ such as

$$\int_{\mathcal{X}} \Omega_1(h(x)(\cdot)) d\mathbf{P}(x) \text{ or } \int_{\Theta} \Omega_2(h(\cdot)(\theta)) d\mu(\theta)$$

  which allow the property enforced by Ω_1 or Ω_2 to hold pointwise on $\mathcal{X}$ or Θ respectively.

It is worthwhile to see a concrete example before turning to the numerical solution (Section 3): in quantile regression, the monotonicity assumption of the $\theta \mapsto h(x)(\theta)$ function can be encoded by choosing Ω_1 as

$$\Omega_1(f) = \lambda_{nc} \int_{\Theta} |-(\partial f)(\theta)|_+ d\mu(\theta).$$

Many different models ($\mathcal{H}$) could be applied to solve this problem. In our work we consider Reproducing Kernel Hilbert Spaces as they offer a simple and principled way to define regularizers by the appropriate choice of kernels and exhibit a significant flexibility.

3 Solving the problem in RKHSs

This section is dedicated to solving the ITL problem defined in Eq. (7). In Section 3.1 we focus on the objective ($\widetilde{V}$). The applied vv-RKHS model family is detailed in Section 3.2 with various penalty examples followed by representer theorems, giving rise to computational tractability.

3.1 Sampled Empirical Risk

In practice solving Eq. (7) can be rather challenging due to the integral over θ. One might consider different numerical integration techniques to handle this issue. We focus here on Quasi Monte Carlo (QMC) methods as they allow (i) efficient optimization over vv-RKHSs which we will use for modelling $\mathcal{H}$ (Proposition 3.1), and (ii) enable us to derive generalization guarantees (Proposition 4.1). Indeed, let

$$\widetilde{V}(y, h(x)) := \sum_{j=1}^{m} w_j \nu(\theta_j, y, h(x)(\theta_j)) \tag{8}$$

be the QMC approximation of Eq. (5). Let $w_j = m^{-1} F^{-1}(\theta_j)$, and $(\theta_j)_{j=1}^m$ be a sequence with values in $[0,1]^d$ such as the Sobol or Halton sequence where μ is assumed to be absolutely continuous w. r. t. the Lebesgue measure and F is the associated cdf. Using this notation and the training samples $S = ((x_i, y_i))_{i=1}^n$, the empirical risk takes the form

$$\widetilde{R}_S(h) := \frac{1}{n} \sum_{i=1}^{n} \widetilde{V}(y_i, h(x_i)) \tag{9}$$

and the problem to solve is

$$\min_{h \in \mathcal{H}} \widetilde{R}_S(h) + \Omega(h). \tag{10}$$

3.2 Hypothesis class ($\mathcal{H}$)

Recall that $\mathcal{H} \subseteq \mathcal{F}(\mathcal{X}; \mathcal{F}(\Theta; \mathcal{Y}))$, in other words $h(x)$ is a $\Theta \mapsto \mathcal{Y}$ function for all $x \in \mathcal{X}$. In this work we assume that the $\Theta \mapsto \mathcal{Y}$ mapping can be described by an RKHS $\mathcal{H}_{k_\Theta}$ associated to a $k_\Theta\colon \Theta \times \Theta \to \mathbb{R}$ scalar-valued kernel defined on the hyperparameters. Let $k_{\mathcal{X}}\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a scalar-valued kernel on the input space. The $x \mapsto$ (hyperparameter $\mapsto$ output)

relation, i. e. $h: \mathcal{X} \to \mathcal{H}_{k_\Theta}$ is then modelled by the Vector-Valued Reproducing Kernel Hilbert Space $\mathcal{H}_K = \overline{\text{span}}\ \{ K(\cdot, x)f \mid \ x \in \mathcal{X},\ \ f \in \mathcal{H}_{k_\Theta} \}$, where the operator-valued kernel K is defined as $K(x, z) = k_\mathcal{X}(x, z)I$, and $I = I_{\mathcal{H}_{k_\Theta}}$ is the identity operator on $\mathcal{H}_{k_\Theta}$.

This so-called decomposable Operator-Valued Kernel has several benefits and gives rise to a function space with a well-known structure. One can consider elements $h \in \mathcal{H}_K$ as mappings from $\mathcal{X}$ to $\mathcal{H}_{k_\Theta}$, and also as functions from $(\mathcal{X} \times \Theta)$ to $\mathbb{R}$. It is indeed known that there is an isometry between $\mathcal{H}_K$ and $\mathcal{H}_{k_\mathcal{X}} \otimes \mathcal{H}_{k_\Theta}$, the RKHS associated to the product kernel $k_\mathcal{X} \otimes k_\Theta$. The equivalence between these views allows a great flexibility and enables one to follow a functional point of view (to analyse statistical aspects) or to leverage the tensor product point of view (to design new kind of penalization schemes). Below we detail various regularizers before focusing on the representer theorems.

- **Ridge Penalty**: For QR and CSC, a natural regularization is the squared vv-RKHS norm

$$\Omega^{\text{RIDGE}}(h) = \tfrac{\lambda}{2}\|h\|^2_{\mathcal{H}_K}, \quad \lambda > 0. \tag{11}$$

  This choice is amenable to excess risk analysis (see Proposition 4.1). It can be also seen as the counterpart of the classical (multi-task regularization term introduced by Sangnier et al. [SFd16], compatible with an infinite number of tasks. $\|\cdot\|^2_{\mathcal{H}_K}$ acts by constraining the solution to a ball of a finite radius within the vv-RKHS, whose shape is controlled by both $k_\mathcal{X}$ and k_Θ.

- **$L^{2,1}$-penalty**: For DLSE, it is more adequate to apply an $L^{2,1}$-RKHS mixed regularizer:

$$\Omega^{\text{DLSE}}(h) = \frac{1}{2}\int_\Theta \|h(\cdot)(\theta)\|^2_{\mathcal{H}_{k_\mathcal{X}}} d\mu(\theta) \tag{12}$$

  which is an example of a Θ-integrated penalty. This Ω choice allows the preservation of the θ-property (see Fig. 2), i. e. that the proportion of the outliers is θ.

- **Shape Constraints**: Taking the example of QR it is advantageous to ensure the monotonicity of the estimated quantile function Let $\partial_\Theta h$ denotes the derivative of $h(x)(\theta)$ with respect to θ. Then one should solve

$$\begin{aligned} &\arg\min_{h \in \mathcal{H}_K} \widetilde{R}_S(h) + \Omega^{\text{RIDGE}}(h) \\ &\text{s. t.} \quad \forall (x, \theta) \in \mathcal{X} \times \Theta, (\partial_\Theta h)(x)(\theta) \geqslant 0. \end{aligned}$$

  However, the functional constraint prevents a tractable optimization scheme. To mitigate this bottleneck, we penalize if the derivative of h w. r. t. θ is negative:

$$\Omega_{\text{nc}}(h) := \lambda_{\text{nc}} \int_\mathcal{X} \int_\Theta |-(\partial_\Theta h)(x)(\theta)|_+ d\mu(\theta) d\mathbf{P}(x). \tag{13}$$

  When $\mathbf{P} := \mathbf{P}_X$ this penalization can rely on the same anchors and weights as the ones used to approximate the integrated loss function:

$$\widetilde{\Omega}_{\text{nc}}(h) = \lambda_{\text{nc}} \sum\nolimits_{i,j=1}^{n,m} w_j |-(\partial_\mathcal{X} h)(x_i)(\theta_j)|_+. \tag{14}$$

  Thus, one can modify the overall regularizer in QR to be

$$\Omega(h) := \Omega^{\text{RIDGE}}(h) + \widetilde{\Omega}_{\text{nc}}(h). \tag{15}$$

3.3 Representer Theorems

Apart from the flexibility of regularizer design, the other advantage of applying vv-RKHS as hypothesis class is that it gives rise to finite-dimensional representation of the ITL solution under mild conditions. The representer theorem Proposition 3.1 applies to CSC when $\lambda_{nc} = 0$ and to QR when $\lambda_{nc} > 0$.

Proposition 3.1 (Representer)**.** *Assume that for $\forall \theta \in \Theta, v_\theta$ is a proper lower semicontinuous convex function with respect to its second argument. Then*

$$\arg\min_{h \in \mathcal{H}_K} \widetilde{R}_S(h) + \Omega(h), \quad \lambda > 0$$

with $\Omega(h)$ defined as in Eq. (15), has a unique solution h^*, and $\exists\ (\alpha_{ij})_{i,j=1}^{n,m}, (\beta_{ij})_{i,j=1}^{n,m} \in \mathbb{R}^{2nm}$ such that $\forall x \in \mathcal{X}$

$$h^*(x) = \sum_{i=1}^{n} k_\mathcal{X}(x, x_i) \left(\sum_{j=1}^{m} \alpha_{ij} k_\Theta(\cdot, \theta_j) + \beta_{ij} (\partial_2 k_\Theta)(\cdot, \theta_j) \right).$$

Sketch of the proof. *First, we prove that the function to minimize is coercive, convex, lower semicontinuous, hence it has a unique minimum. Then $\mathcal{H}_K$ is decomposed into two orthogonal subspaces and we use the reproducing property to get the finite representation.*

For DLSE, we similarly get a representer theorem with the following modelling choice. Let $k_b : \Theta \times \Theta \to \mathbb{R}$ be a scalar-valued kernel (possibly different from k_θ), $\mathcal{H}_{k_b}$ the associated RKHS and $t \in \mathcal{H}_{k_b}$. Assume also that $\Theta \subseteq [\epsilon, 1]$ where $\epsilon > 0$.* Then, learning a continuum of level sets boils down to the minimization problem

$$\arg\min_{h \in \mathcal{H}_K, t \in \mathcal{H}_{k_b}} \widetilde{R}_S(h, t) + \widetilde{\Omega}(h, t), \quad \lambda > 0, \tag{16}$$

*We choose $\Theta \subseteq [\epsilon, 1]$, $\epsilon > 0$ rather than $\Theta \subseteq [0, 1]$ because the loss might not be integrable on $[0, 1]$.

where $\widetilde{\Omega}(h,t) = \frac{1}{2}\sum_{j=1}^{m} w_j \|h(\cdot)(\theta_j)\|^2_{\mathcal{H}_{k_\mathcal{X}}} + \frac{\lambda}{2}\|t\|^2_{\mathcal{H}_{k_b}}$, $\widetilde{R}_S(h,t) = \frac{1}{n}\sum_{i,j=1}^{n,m} \frac{w_j}{\theta_j}\left(|t(\theta_j) - h(x_i)(\theta_j)|_+ - t(\theta_j)\right)$.

Proposition 3.2 (Representer)**.** *Assume that* k_Θ *is bounded:* $\sup_{\theta\in\Theta} k_\Theta(\theta,\theta) < +\infty$*. Then the minimization problem described in Eq. (*16*) has a unique solution* (h^*, t^*) *and there exist* $(\alpha_{ij})_{i,j=1}^{n,m} \in \mathbb{R}^{n\times m}$ *and* $(\beta_j)_{j=1}^{m} \in \mathbb{R}^m$ *such that for* $\forall (x,\theta) \in \mathcal{X}\times[\epsilon, 1]$*,*

$$h^*(x)(\theta) = \sum_{i,j=1}^{n,m} \alpha_{ij} k_\mathcal{X}(x, x_i) k_\Theta(\theta, \theta_j),$$
$$t^*(\theta) = \sum_{j=1}^{m} \beta_j k_b(\theta, \theta_j).$$

Sketch of the proof. *First we show that the infimum exists, and that it must be attained in some subspace of* $\mathcal{H}_K \times \mathcal{H}_{k_b}$ *over which the objective function is coercive. By the reproducing property, we get the claimed finite decomposition.*

Remarks:

- Models with bias: it can be advantageous to add a bias to the model, which is here a function of the hyperparameter θ: $h(x)(\theta) = f(x)(\theta) + b(\theta)$, $f \in \mathcal{H}_K$, $b \in \mathcal{H}_{k_b}$, where $k_b : \Theta\times\Theta \to \mathbb{R}$ is a scalar-valued kernel. This can be the case for example if the kernel on the hyperparameters is the constant kernel, i. e. $k_\Theta(\theta,\theta') = 1$ ($\forall \theta,\theta' \in \Theta$), hence the model $f(x)(\theta)$ would not depend on θ. An analogous statement to Proposition 3.1 still holds for the biased model if one adds a regularization $\lambda_b \|b\|^2_{\mathcal{H}_{k_b}}$, $\lambda_b > 0$ to the risk.
- Relation to JQR: In ∞-QR, by choosing k_Θ to be the Gaussian kernel, $k_b(x,z) = \mathbb{1}_{\{x\}}(z)$, $\mu = \frac{1}{m}\sum_{j=1}^{m}\delta_{\theta_j}$, where δ_θ is the Dirac measure concentrated on θ, one gets back Sangnier et al. [SFd16]'s Joint Quantile Regression (JQR) framework as a special case of our approach. In contrast to the JQR, however, in ∞-QR one can predict the quantile value at any $\theta \in (0,1)$, even outside the $(\theta_j)_{j=1}^m$ used for learning.
- Relation to q-OCSVM: In DLSE, by choosing $k_\Theta(\theta,\theta') = 1$ (for all $\theta,\theta' \in \Theta$) to be the constant kernel, $k_b(\theta,\theta') = \mathbb{1}_{\{\theta\}}(\theta')$, $\mu = \frac{1}{m}\sum_{j=1}^{m}\delta_{\theta_j}$, our approach specializes to q-OCSVM [GLM13].
- Relation to Kadri et al. [Kad+16]: Note that Operator-Valued Kernels for functional outputs have also been used in [Kad+16], under the form of integral operators acting on L^2 spaces. Both kernels give rise to the same space of functions; the benefit of our approach being to provide an *exact* finite representation of the solution (see Proposition 3.1).
- Efficiency of the decomposable kernel: this kernel choice allows to rewrite the expansions in Propositions 3.1 and 3.2 as a Kronecker products and the complexity of the prediction of n' points for m' quantile becomes $\mathcal{O}(m'mn + n'nm)$ instead of $\mathcal{O}(m'mn'n)$.

4 Excess Risk Bounds

Below we provide a generalization error analysis to the solution of Eq. (10) for QR and CSC (with Ridge regularization and without shape constraints) by stability argument [BE02], extending the work of Audiffren et al. [AK13] to Infinite-Task Learning. The proposition instantiates the guarantee for the QMC scheme.

Proposition 4.1 (Generalization)**.** *Let* $h^* \in \mathcal{H}_K$ *be the solution of Eq. (*10*) for the* QR *or* CSC *problem with* QMC *approximation. Under mild conditions on the kernels* $k_\mathcal{X}, k_\Theta$ *and* $\mathbf{P}_{X,Y}$*, one has*

$$R(h^*) \leqslant \widetilde{R}_S(h^*) + \mathcal{O}_{\mathbf{P}_{X,Y}}\left(\frac{1}{\sqrt{\lambda n}}\right) + \mathcal{O}\left(\frac{\log(m)}{\sqrt{\lambda} m}\right). \quad (17)$$

Sketch of the proof. *The error resulting from sampling* $\mathbf{P}_{X,Y}$ *and the inexact integration is respectively bounded by* β*-stability [Kad+16] and* QMC *results.*[†]

(n, m) Trade-off: The proposition reveals the interplay between the two approximations, n (the number of training samples) and m (the number of locations taken in the integral approximation), and allows to identify the regime in $\lambda = \lambda(n,m)$ driving the excess risk to zero. Indeed by choosing $m = \sqrt{n}$ and discarding logarithmic factors for simplicity, $\lambda \gg n^{-1}$ is sufficient. The mild assumptions imposed are: boundedness on both kernels and the random variable Y, as well as some smoothness of the kernels.

5 Numerical Examples

In this section we provide numerical examples illustrating the efficiency of the proposed ITL approach.[‡] We used the following datasets in our experiments:

- Quantile Regression: we used (i) a sine synthetic benchmark [SFd16]: a sine curve at 1Hz modulated by a sine envelope at 1/3Hz and mean 1, distorted with a Gaussian noise of mean 0 and a linearly decreasing standard deviation from 1.2 at $x = 0$ to 0.2 at $x = 1.5$.

[†]The QMC approximation may involve the Sobol sequence with discrepancy $m^{-1}\log(m)^s$ ($s = \dim(\Theta)$).

[‡]The code is available at `https://bitbucket.org/RomainBrault/itl`.

Table 1: Quantile Regression on 20 UCI datasets. Reported: 100×value of the pinball loss, 100×crossing loss (smaller is better). p.-val.: outcome of the Mann-Whitney-Wilcoxon test of JQR vs. ∞-QR and Independent vs. ∞-QR. Boldface: significant values w. r. t. ∞-QR.

DATASET	JQR				IND-QR				∞-QR	
	(PINBALL	P.-VAL.)	(CROSS	P.-VAL.)	(PINBALL	P.-VAL.)	(CROSS	P.-VAL.)	PINBALL	CROSS
COBARORE	159 ± 24	$9 \cdot 10^{-01}$	0.1 ± 0.4	$6 \cdot 10^{-01}$	150 ± 21	$2 \cdot 10^{-01}$	0.3 ± 0.8	$7 \cdot 10^{-01}$	165 ± 36	2.0 ± 6.0
ENGEL	175 ± 555	$6 \cdot 10^{-01}$	0.0 ± 0.2	$1 \cdot 10^{+00}$	63 ± 53	$8 \cdot 10^{-01}$	4.0 ± 12.8	$8 \cdot 10^{-01}$	47 ± 6	0.0 ± 0.1
BOSTONHOUSING	49 ± 4	$8 \cdot 10^{-01}$	0.7 ± 0.7	$2 \cdot 10^{-01}$	49 ± 4	$8 \cdot 10^{-01}$	$\mathbf{1.3 \pm 1.2}$	$1 \cdot 10^{-05}$	49 ± 4	0.3 ± 0.5
CAUTION	88 ± 17	$6 \cdot 10^{-01}$	0.1 ± 0.2	$6 \cdot 10^{-01}$	89 ± 19	$4 \cdot 10^{-01}$	$\mathbf{0.3 \pm 0.4}$	$2 \cdot 10^{-04}$	85 ± 16	0.0 ± 0.1
FTCOLLINSSNOW	154 ± 16	$8 \cdot 10^{-01}$	0.0 ± 0.0	$6 \cdot 10^{-01}$	155 ± 13	$9 \cdot 10^{-01}$	0.2 ± 0.9	$8 \cdot 10^{-01}$	156 ± 17	0.1 ± 0.6
HIGHWAY	103 ± 19	$4 \cdot 10^{-01}$	0.8 ± 1.4	$2 \cdot 10^{-02}$	99 ± 20	$9 \cdot 10^{-01}$	$\mathbf{6.2 \pm 4.1}$	$1 \cdot 10^{-07}$	105 ± 36	0.1 ± 0.4
HEIGHTS	127 ± 3	$1 \cdot 10^{+00}$	0.0 ± 0.0	$1 \cdot 10^{+00}$	127 ± 3	$9 \cdot 10^{-01}$	0.0 ± 0.0	$1 \cdot 10^{+00}$	127 ± 3	0.0 ± 0.0
SNIFFER	43 ± 6	$8 \cdot 10^{-01}$	0.1 ± 0.3	$2 \cdot 10^{-01}$	44 ± 5	$7 \cdot 10^{-01}$	$\mathbf{1.4 \pm 1.2}$	$6 \cdot 10^{-07}$	44 ± 7	0.1 ± 0.1
SNOWGEESE	55 ± 20	$7 \cdot 10^{-01}$	0.3 ± 0.8	$3 \cdot 10^{-01}$	53 ± 18	$6 \cdot 10^{-01}$	0.4 ± 1.0	$5 \cdot 10^{-02}$	57 ± 20	0.2 ± 0.6
UFC	81 ± 5	$6 \cdot 10^{-01}$	$\mathbf{0.0 \pm 0.0}$	$4 \cdot 10^{-04}$	82 ± 5	$7 \cdot 10^{-01}$	$\mathbf{1.0 \pm 1.4}$	$2 \cdot 10^{-04}$	82 ± 4	0.1 ± 0.3
BIGMAC2003	80 ± 21	$7 \cdot 10^{-01}$	$\mathbf{1.4 \pm 2.1}$	$4 \cdot 10^{-04}$	74 ± 24	$9 \cdot 10^{-02}$	$\mathbf{0.9 \pm 1.1}$	$7 \cdot 10^{-05}$	84 ± 24	0.2 ± 0.4
UN3	98 ± 9	$8 \cdot 10^{-01}$	0.0 ± 0.0	$1 \cdot 10^{-01}$	99 ± 9	$1 \cdot 10^{+00}$	$\mathbf{1.2 \pm 1.0}$	$1 \cdot 10^{-05}$	99 ± 10	0.1 ± 0.4
BIRTHWT	141 ± 13	$1 \cdot 10^{+00}$	0.0 ± 0.0	$6 \cdot 10^{-01}$	140 ± 12	$9 \cdot 10^{-01}$	0.1 ± 0.2	$7 \cdot 10^{-02}$	141 ± 12	0.0 ± 0.0
CRABS	$\mathbf{11 \pm 1}$	$4 \cdot 10^{-05}$	0.0 ± 0.0	$8 \cdot 10^{-01}$	$\mathbf{11 \pm 1}$	$2 \cdot 10^{-04}$	$\mathbf{0.0 \pm 0.0}$	$2 \cdot 10^{-05}$	13 ± 3	0.0 ± 0.0
GAGURINE	61 ± 7	$4 \cdot 10^{-01}$	0.0 ± 0.1	$3 \cdot 10^{-03}$	62 ± 7	$5 \cdot 10^{-01}$	$\mathbf{0.1 \pm 0.2}$	$4 \cdot 10^{-04}$	62 ± 7	0.0 ± 0.0
GEYSER	105 ± 7	$9 \cdot 10^{-01}$	0.1 ± 0.3	$9 \cdot 10^{-01}$	105 ± 6	$9 \cdot 10^{-01}$	0.2 ± 0.3	$6 \cdot 10^{-01}$	104 ± 6	0.1 ± 0.2
GILGAIS	51 ± 6	$5 \cdot 10^{-01}$	0.1 ± 0.1	$1 \cdot 10^{-01}$	49 ± 6	$6 \cdot 10^{-01}$	$\mathbf{1.1 \pm 0.7}$	$2 \cdot 10^{-05}$	49 ± 7	0.3 ± 0.3
TOPO	69 ± 18	$1 \cdot 10^{+00}$	0.1 ± 0.5	$1 \cdot 10^{+00}$	71 ± 20	$1 \cdot 10^{+00}$	$\mathbf{1.7 \pm 1.4}$	$3 \cdot 10^{-07}$	70 ± 17	0.0 ± 0.0
MCYCLE	66 ± 9	$9 \cdot 10^{-01}$	0.2 ± 0.3	$7 \cdot 10^{-03}$	66 ± 8	$9 \cdot 10^{-01}$	$\mathbf{0.3 \pm 0.3}$	$7 \cdot 10^{-06}$	65 ± 9	0.0 ± 0.1
CPUS	$\mathbf{7 \pm 4}$	$2 \cdot 10^{-04}$	$\mathbf{0.7 \pm 1.0}$	$5 \cdot 10^{-04}$	$\mathbf{7 \pm 5}$	$3 \cdot 10^{-04}$	$\mathbf{1.2 \pm 0.8}$	$6 \cdot 10^{-08}$	16 ± 10	0.0 ± 0.0

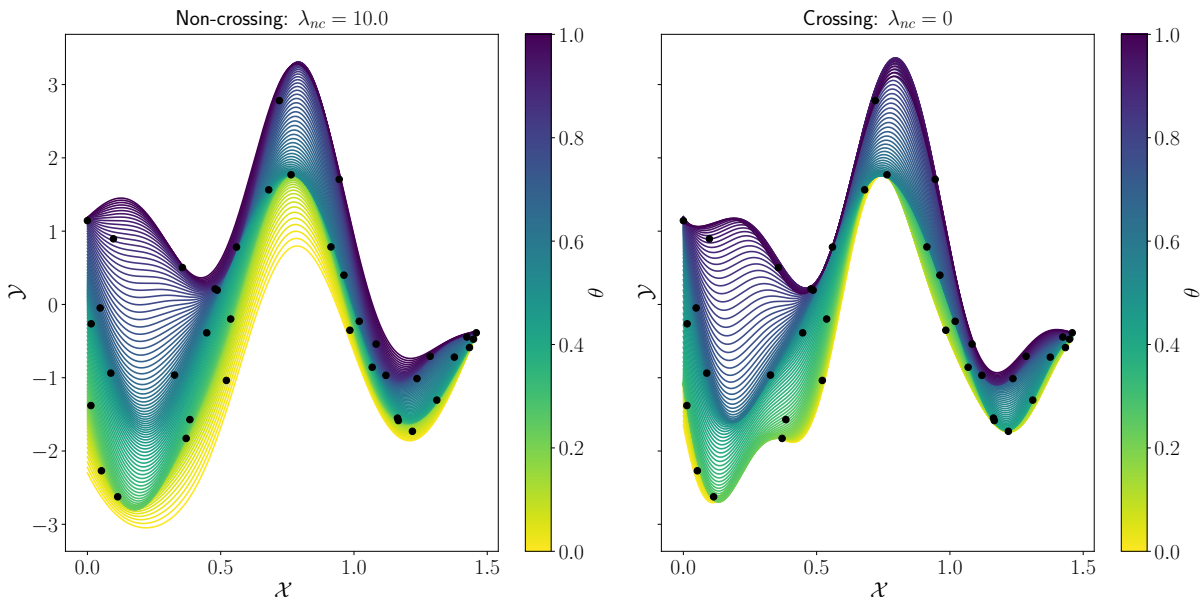


Figure 1: Impact of crossing penalty on toy data. Left plot: strong non-crossing penalty ($\lambda_{\text{nc}} = 10$). Right plot: no non-crossing penalty ($\lambda_{\text{nc}} = 0$). The plots show 100 quantiles of the continuum learned, linearly spaced between 0 (blue) and 1 (red). Notice that the non-crossing penalty does not provide crossings to occur in the regions where there is no points to enforce the penalty (e. g. $x \in [0.13, 0.35]$). This phenomenon is alleviated by the regularity of the model.

(ii) 20 standard regression datasets from UCI. The number of samples varied between 38 (CobarOre) and 1375 (Height). The observations were standardised to have unit variance and zero mean for each attribute.

- Density Level-Set Estimation: The Wilt database from the UCI repository with 4839 samples and 5 attributes, and the Spambase UCI dataset with 4601 samples and 57 attributes served as benchmarks.

Note on Optimization: There are several ways to solve the non-smooth optimization problems associated to the QR, DLSE and CSC tasks. One could proceed for example by duality—as it was done in JQR Sangnier et al. [SFd16]—, or apply sub-gradient descent techniques (which often converge quite slowly). In order to allow unified treatment and efficient solution in our experiments we used the L-BFGS-B [Zhu+97] optimization scheme which is widely popular in large-scale learning, with non-smooth extensions [KW17]. The technique requires only evaluation of objective function along with its gradient, which can be computed automatically using reverse mode automatic differentiation (as in Abadi et al. [Aba+16]). To benefit from from the available fast smooth implementations [JOP+01], we applied an infimal convolution on the non-differentiable terms of the objective. Under the assumtion that $m = \mathcal{O}(\sqrt{n})$ (see Proposition 4.1), the complexity per L-BFGS-B iteration is $\mathcal{O}(n^2\sqrt{n})$.

QR: The efficiency of the non-crossing penalty is illustrated in Fig. 1 on the synthetic sine wave dataset described in Section 5 where $n = 40$ and $m = 20$ points have been generated. Many crossings are visible on the right plot, while they are almost not noticible on the left plot, using the non-crossing penalty. Concerning our real-world examples, to study the efficiency of the proposed scheme in quantile regression the following experimental protocol was applied. Each dataset (Section 5) was splitted randomly into a training set (70%) and a test set (30%). We optimized the hyperparameters by minimizing a 5-folds cross validation with a Bayesian optimizer[§]. Once the hyperparameters were obtained, a new regressor was learned on the whole training set using the optimized hyperparameters. We report the value of the pinball loss and the crossing loss on the test set for three methods: our technique is called ∞-QR, we refer to Sangnier et al. [SFd16]'s approach as JQR, and independent learning (abbreviated as IND-QR) represents a further baseline.

We repeated 20 simulations (different random training-test splits); the results are also compared using a Mann-Whitney-Wilcoxon test. A summary is provided in Table 1. Notice that while JQR is tailored to predict finite many quantiles, our ∞-QR method estimates the *whole quantile function* hence solves a more challenging task. Despite the more difficult problem solved, as Table 1 suggest that the performance in terms of pinball loss of ∞-QR is comparable to that of the state-of-the-art JQR on all the twenty studied benchmarks, except for the 'crabs' and 'cpus' datasets (p.-val. $< 0.25\%$). In addition, when considering the non-crossing penalty one can observe that ∞-QR outperforms the IND-QR baseline on eleven datasets (p.-val. $< 0.25\%$) and JQR on two datasets. This illustrates the efficiency of the constraint based on the continuum scheme.

CSC: To illustrate the advantage of (infinite) joint learning we used two synthetic datasets Circles and Two-Moons and the UCI Iris dataset. We chose $k_{\mathcal{X}}$ to be a Gaussian kernel with bandwidth $\sigma_{\mathcal{X}} = (2\gamma_{\mathcal{X}})^{(-1/2)}$ the median of the Euclidean pairwise distances of the input points [JDH99]. k_{Θ} is also a Gaussian kernel with bandwidth $\gamma_{\Theta} = 5$. We used $m = 20$ for all datasets. As a baseline we trained independently 3 Cost-Sensitive Classification classifiers with $\theta \in \{-0.9, 0, 0.9\}$. We repeated 50 times a random $50 - 50\%$ train-test split of the dataset and report the average test error and standard deviation (in terms of sensitivity and specificity)

Our results are illustrated in Table 2. For $\theta = -0.9$, both independent and joint learners give the desired 100% specificity; the joint Cost-Sensitive Classification scheme however has significantly higher sensitivity value (15% vs 0%) on the dataset Circles. Similar conclusion holds for the $\theta = +0.9$ extreme: the ideal sensitivity is reached by both techniques, but the joint learning scheme performs better in terms of specificity (0% vs 12%) on the dataset Circles.

DLSE: To assess the quality of the estimated model by ∞-OCSVM, we illustrate the θ-property [Sch+00]: the proportion of inliers has to be approximately $1 - \theta$ ($\forall \theta \in (0, 1)$). For the studied datasets (Wilt, Spambase) we used the raw inputs without applying any preprocessing. Our input kernel was the exponentiated χ^2 kernel $k_{\mathcal{X}}(x, z) := \exp\left(-\gamma_{\mathcal{X}} \sum_{k=1}^{d} (x_k - z_k)^2/(x_k + z_k)\right)$ with bandwidth $\gamma_{\mathcal{X}} = 0.25$. A Gauss-Legendre quadrature rule provided the integral approximation in Eq. (8),

[§]We used a Gaussian Process model and minimized the Expected improvement. The optimizer was initialized using 27 samples from a Sobol sequence and ran for 50 iterations.

Table 2: ∞-CSC vs Independent (IND)-CSC. Higher is better.

Dataset	Method	$\theta = -0.9$		$\theta = 0$		$\theta = +0.9$	
		sensitivity	specificity	sensitivity	specificity	sensitivity	specificity
Two-Moons	IND	0.3 ± 0.05	0.99 ± 0.01	0.83 ± 0.03	0.86 ± 0.03	0.99 ± 0	0.32 ± 0.06
	∞-CSC	0.32 ± 0.05	0.99 ± 0.01	0.84 ± 0.03	0.87 ± 0.03	1 ± 0	0.36 ± 0.04
Circles	IND	0 ± 0	1 ± 0	0.82 ± 0.02	0.84 ± 0.03	1 ± 0	0 ± 0
	∞-CSC	0.15 ± 0.05	1 ± 0	0.82 ± 0.02	0.84 ± 0.03	1 ± 0	0.12 ± 0.05
Iris	IND	0.88 ± 0.08	0.94 ± 0.06	0.94 ± 0.05	0.92 ± 0.06	0.97 ± 0.05	0.87 ± 0.06
	∞-CSC	0.89 ± 0.08	0.94 ± 0.05	0.94 ± 0.06	0.92 ± 0.05	0.97 ± 0.04	0.90 ± 0.05
Toy	IND	0.51 ± 0.06	0.98 ± 0.01	0.83 ± 0.03	0.86 ± 0.03	0.97 ± 0.01	0.49 ± 0.07
	∞-CSC	0.63 ± 0.04	0.96 ± 0.01	0.83 ± 0.03	0.85 ± 0.03	0.95 ± 0.02	0.61 ± 0.04

with $m = 100$ samples. We chose the Gaussian kernel for k_Θ; its bandwidth parameter γ_Θ was the 0.2−quantile of the pairwise Euclidean distances between the θ_j's obtained via the quadrature rule. The margin (bias) kernel was $k_b = k_\Theta$. As it can be seen in Fig. 2, the θ-property holds for the estimate which illustrates the efficiency of the proposed continuum approach for density level-set estimation.

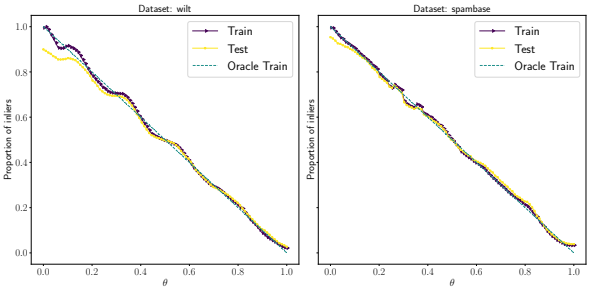


Figure 2: Density Level-Set Estimation: the θ-property is approximately satisfied.

6 Conclusion

In this work we proposed Infinite Task Learning, a novel nonparametric framework aiming at jointly solving parametrized tasks for a continuum of hyperparameters. We provided excess risk guarantees for the studied ITL scheme, and demonstrated its practical efficiency and flexibility in various tasks including cost-sensitive classification, quantile regression and density level set estimation.

Acknowledgments

The authors thank Arthur Tenenhaus for some insightful discussions. This work was supported by the Labex *DigiCosme* (project ANR-11-LABEX-0045-DIGICOSME) and the industrial chair *Machine Learning for Big Data* at Télécom ParisTech.

References

[Aba+16] M. Abadi et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2016, pp. 265–283 (cit. on p. 8).

[AK13] J. Audiffren and H. Kadri. "Stability of Multi-Task Kernel Regression Algorithms". In: *Asian Conference on Machine Learning (ACML)*. Vol. 29. PMLR, 2013, pp. 1–16 (cit. on p. 6).

[ÁRL12] M. A. Álvarez, L. Rosasco, and N. D. Lawrence. "Kernels for vector-valued functions: a review". In: *Foundations and Trends in Machine Learning* 4.3 (2012), pp. 195–266 (cit. on p. 3).

[Bac+12] F. Bach et al. "Optimization with sparsity-inducing penalties". In: *Foundations and Trends in Machine Learning* 4.1 (2012), pp. 1–106 (cit. on p. 1).

[BE02] O. Bousquet and A. Elisseeff. "Stability and generalization". In: *Journal of Machine Learning Research* 2 (2002), pp. 499–526 (cit. on p. 6).

[Car+10] C. Carmeli et al. "Vector valued reproducing kernel Hilbert spaces and universality". In: *Analysis and Applications* 8 (1 2010), pp. 19–61 (cit. on p. 2).

[Cil+17] C. Ciliberto et al. "Consistent multitask learning with nonlinear output relations". In: *Advances in Neural Information Processing Systems (NIPS)*. 2017, pp. 1986–1996 (cit. on p. 3).

[EMP05] T. Evgeniou, C. A. Micchelli, and M. Pontil. "Learning Multiple Tasks with kernel methods". In: *JMLR* 6 (2005), pp. 615–637 (cit. on p. 3).

[EP04] T. Evgeniou and M. Pontil. "Regularized multi–task learning". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, pp. 109–117 (cit. on p. 1).

[GLM13] A. Glazer, M. Lindenbaum, and S. Markovitch. "q-OCSVM: A q-quantile estimator for high-dimensional distributions". In: *Advances in Neural Information Processing Systems (NIPS)*. 2013, pp. 503–511 (cit. on pp. 1–3, 6).

[Has+04] T. Hastie et al. "The entire regularization path for the support vector machine". In: *Journal of Machine Learning Research* 5.Oct (2004), pp. 1391–1415 (cit. on p. 1).

[JDH99] T. Jaakkola, M. Diekhans, and D. Haussler. "Using the Fisher kernel method to detect remote protein homologies." In: *ISMB*. Vol. 99. 1999, pp. 149–158 (cit. on p. 8).

[JOP+01] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001 (cit. on p. 8).

[Kad+16] H. Kadri et al. "Operator-valued Kernels for Learning from Functional Response Data". In: *Journal of Machine Learning Research* 17 (2016), pp. 1–54 (cit. on p. 6).

[KB78] R. Koenker and G. Bassett Jr. "Regression quantiles". In: *Econometrica: journal of the Econometric Society* (1978), pp. 33–50 (cit. on pp. 1, 2).

[KW17] N. Keskar and A. Wächter. "A limited-memory quasi-Newton algorithm for bound-constrained non-smooth optimization". In: *Optimization Methods and Software* (2017), pp. 1–22 (cit. on p. 8).

[Ped57] G. Pedrick. "Theory of reproducing kernels for Hilbert spaces of vector-valued functions". PhD thesis. University of Kansas, 1957 (cit. on p. 1).

[Sch+00] B. Schölkopf et al. "New support vector algorithms". In: *Neural computation* 12.5 (2000), pp. 1207–1245 (cit. on pp. 1, 3, 8).

[SFd16] M. Sangnier, O. Fercoq, and F. d'Alché-Buc. "Joint quantile regression in vector-valued RKHSs". In: *Advances in Neural Information Processing Systems (NIPS)* (2016), pp. 3693–3701 (cit. on pp. 2, 3, 5, 6, 8).

[TA77] A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill-posed Problems*. Winston & Sons, 1977 (cit. on p. 1).

[Tak+06] I. Takeuchi et al. "Nonparametric quantile estimation". In: *Journal of Machine Learning Research* 7 (2006), pp. 1231–1264 (cit. on p. 1).

[Tak+13] I. Takeuchi et al. "Parametric task learning". In: *Advances in Neural Information Processing Systems (NIPS)*. 2013, pp. 1358–1366 (cit. on pp. 1, 3).

[Tib96] R. Tibshirani. "Regression shrinkage and selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288 (cit. on p. 1).

[VGS97] V. Vapnik, S. E. Golowich, and A. J. Smola. "Support vector method for function approximation, regression estimation and signal processing". In: *Advances in Neural Information Processing Systems (NIPS)*. 1997, pp. 281–287 (cit. on p. 1).

[VV06] R. Vert and J.-P. Vert. "Consistency and convergence rates of one-class SVMs and related algorithms". In: *Journal of Machine Learning Research* 7 (2006), pp. 817–854 (cit. on p. 3).

[ZE01] B. Zadrozny and C. Elkan. "Learning and making decisions when costs and probabilities are both unknown". In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2001, pp. 204–213 (cit. on p. 1).

[Zhu+97] C. Zhu et al. "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization". In: *ACM Transactions on Mathematical Software (TOMS)* 23.4 (1997), pp. 550–560 (cit. on p. 8).

A proposal for adversarially robust vision models: pairing unsupervised generative objectives with supervised semantic labels.

Bhavin Choksi[1], Shikhar Seth[1], Andrea Alamia[1], Benjamin Ador[1], and Rufin VanRullen*[1]

[1]CerCo, CNRS UMR5549, Université de Toulouse

June 1, 2019

Abstract

Neural networks have greatly influenced the field of machine learning in the past decade, becoming the first method of choice in the presence of "big data". However, it was recently shown that these networks are susceptible to so-called *adversarial examples*: small changes to the inputs resulting in completely different (and inaccurate) outputs. After a brief review of adversarial attacks and defenses, we propose a two-step paradigm for building robust neural networks in the domain of computer vision. First, the network must learn to retain inputs onto the data manifold (the low-dimensional subset of all possible inputs defined by the training dataset); inputs deviating from this manifold must be projected back onto it. This property can be achieved via unsupervised training of a generative model. Second, we propose that supervised training taking into account the semantic proximity between classes can further improve the network robustness by restructuring the manifold in a more meaningful way. Previous studies have already found that each of these two components in isolation can enhance network robustness or performance, but they have never been combined into a single model. We propose a novel deep neural network implementation, motivated by the architecture of the primate visual system, that combines unsupervised "predictive coding" to achieve step 1 (generative model) and supervised training based on an NLP (Natural Language Processing) word-embedding space to achieve step 2 (semantic proximity).

Keywords: Neural networks, adversarial examples, auto-encoders, predictive coding, word embedding.

*Corresponding author: rufin.vanrullen@cnrs.fr

1 Introduction

Deep neural networks (DNNs) have revolutionized machine learning approaches in the past decade. In computer vision, they have shown near- or even supra-human accuracy in classifying categories from complex datasets like ImageNet [1, 2].

Despite the widespread adoption of DNNs, in 2013 Szegedy et al. discovered that, unlike humans, these neural networks are sensitive to minute perturbations to the input image – so called 'adversarial perturbations' [3]. For example, a small change in the pixel values of an image of a "truck" would make the model classify it as an "ostrich" with a high confidence.

This discovery motivated various studies inquiring as to what caused such sensitivity in neural architectures [4, 5, 6, 7]. It also posed a threat to the general use of such systems. For example, one could easily fool a self-driving system to misjudge the safety signs on the road or distribute a malware disguised as a benign software [8]. Deploying such vulnerable machine learning systems would allow ill-intentioned parties to exploit them to commit frauds. Building robust models is therefore crucial to ensure general safety. We briefly summarize the current understanding of the field in section 2, develop our hypotheses in section 3, and propose one specific implementation in section 4.

2 Background: a primer on adversarial attacks and defenses

Notation: Suppose a Classifier C is trained on an image dataset $\mathbb{D}$ containing examples $(x_i, y_i) \in \mathbb{D}$ such that $C(x_i; \theta) = y_i$, where y_i denotes the true label

of the image x_i and θ denotes the underlying trained parameters of the model. Then, an adversarial example $\tilde{x}_i$ is an image, such that,

$$C(\tilde{x}_i; \theta) = y_j$$

where $y_j \neq y_i$ and $d(\tilde{x}_i, x_i) \leq \epsilon$ where d is some distance metric and ϵ is an arbitrary small change in that metric.

A typical choice for the distance metric d is some l_p-norm with a suitable value of p. For instance, in l_2-norm, the mean-squared distance between the images has to be less than ϵ. Similarly, in case of l_∞-norm as the constraint, any change in the pixel values of an image cannot be greater than ϵ. The motivation to keep ϵ low is to keep the adversarial image as close as possible to the original image for the human eye [9, 10].

Types of adversarial attacks: Adversarial attacks can be broadly classified into two types - targeted and untargeted. Targeted attacks are those whose aim is to force a model to predict a specific label ($y_j = y_t$ where y_t is a target label $\neq y_i$). On the other hand, untargeted attacks are designed to force the model to misclassify the input image regardless of the final output label (y_j is any label other than y_i) [11, 10, 9]

Furthermore, adversarial attacks can be distinguished along another axis, i.e. considering the information the attacker has about the targeted model. If complete information about the model's architecture is available (and in particular, the error gradients), such attacks are labeled as 'white-box' attacks. Contrarily, if no information is available, the attacks are labeled as 'black-box' attacks [11, 12, 10]. If the attacker has access to only a certain amount of information, then the adversarial attacks are defined as 'grey-box' attacks [13].

To each attack (e.g. targeted or untargeted) corresponds a success criterion, e.g. $C(\tilde{x}_i; \theta) = y_t$, which can also be expressed as an appropriate loss function $L_{adv}(.)$. The aim of the attacker is then to minimize this adversarial loss function by making minimal perturbations δ to the original image.

$$\min \Big[L_{adv}(x_i + \delta) \Big] \text{such that } ||\delta||_p \leq \epsilon$$

For instance, the Fast Gradient Sign Method (FGSM) [14] aims to move towards the adversarial criterion along the gradient of the loss function with a step-size of α:

$$\tilde{x}_i = x_i - \alpha sign(\nabla L_{adv}(x_i))$$

Attack methods: Similar to FGSM, a suite of attacks including Basic Iterative Method (BIM), Jacobian-based saliency map (JSMA) or Carlini-Wagner attacks has been developed which utilize the gradients of the model to generate adversarial examples [14, 7, 10, 15].

Another set of attacks use the 'transferability' property of the adversarial examples. It has been observed that an adversarial image against one model has a higher probability of being adversarial against another model. In transfer-based attacks, the attacker develops adversarial perturbations against undefended models (e.g. models for which the gradients are available) and uses them against the target model (potentially one for which gradients or other variables are obfuscated) [11].

A final class of attacks is called 'decision-based' attacks. This includes a recently developed model-agnostic (black-box) approach, called 'boundary attack', which uses only the hard-labels given by the mode to generate adversarial images [16]. This decision-based iterative attack starts from an adversarial point and moves closer to the original image on each step, while still remaining adversarial. This can be done without knowledge of or access to the underlying architecture or parameters of the target model.

Defense methods: Szegedy et al (2013) [3] suggested augmenting the training dataset with the adversarially perturbed images to increase the robustness of the models, a method known as 'adversarial training'. Madry et al showed that if models are adversarially trained with perturbations that maximize the model's loss, then they can be made robust to white-box attacks [5]. To this day, adversarial training remains one of the most powerful defense methods.

To defend models against transfer-based attacks, Tramer et al. [17] designed a similar strategy, which augmented the training dataset with the adversarial examples generated for another model. This strategy, named as *ensemble adversarial training* proved successful in defeating most of the transfer-based attacks.

Gradient-based attacks like FGSM, BIM, JSMA, etc. were shown to be foiled by masking the gradients of the model using various forms of non-linearities (so-called 'obfuscated gradients' methods [12, 18, 19]). Later on though, new attacks were developed which only estimated (instead of computing) the gradients, and could successfully break those defenses [12]. In addition, black-box attacks such as 'boundary attacks', which rely only on the model decision and not its gradients, remain unaffected by these 'obfuscated gradients' and can easily break these defenses.

One interpretation of adversarial attack and defense

methods, illustrated in Figure 1, is that adversarial perturbations often take the image off the data manifold, where the model has very little (if any) training, thus making it easier to fool the model to give an incorrect label to the image. Adversarial training, in this context, attempts to defend the model by increasing the density of data-points on-and-around the manifold itself. Adversarial training, however, is a computationally costly method to achieve robustness, because several new adversarial examples must be generated for each training sample. A possibly simpler strategy could be to use generative models that learn to project their (potentially adversarial) inputs back onto the initial data manifold (see our Hypothesis (1) in Section 3). Several recent papers have used this approach to improve adversarial robustness. They mainly differ with respect to the generative model used to retain (potentially adversarial) inputs onto the manifold: using PixelCNN or PixelRNN for the 'PixelDefend' method [6], GANs for the 'DefenseGAN' [20] and 'APE-GAN' [21] methods, auto-encoders for the 'MagNet' [22] or 'Robust Manifold' [23] methods, sparse coding [24], predictive coding [25] or image super-resolution [26].

All these generative defense methods clearly highlight the importance of projecting the image back onto the manifold before performing classification. Unfortunately, these defenses were found to protect only partially against adversarial attacks: even on the manifold, adversarial perturbations can still be found [23, 12]. The limited success of these generative defense methods hints that a mere projection, although necessary, is not sufficient to provide a foolproof solution to this problem.

This brings us to our hypothesis.

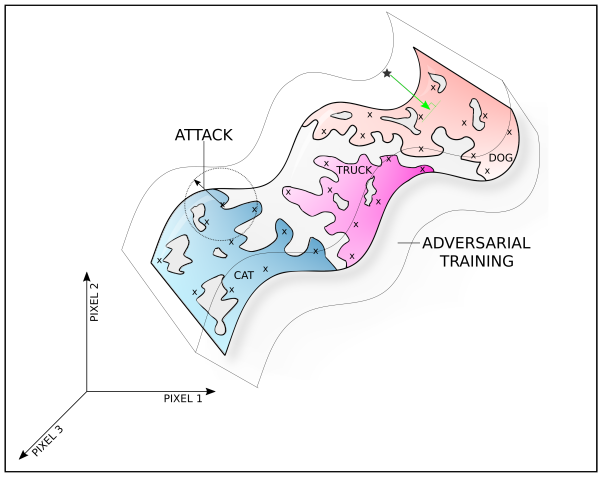


Figure 1: **A 'manifold' view on adversarial attacks and defenses.** A training dataset (such as ImageNet) defines a lower-dimensional sub-space of the full input space corresponding to all possible pixel values. This sub-space is the 'data manifold', illustrated here as a 2D sheet in a 3D pixel space (in reality, the space has many more dimensions, and the data manifold is typically much less smooth). During supervised learning, classifiers learn to associate labels (here, colors) to training samples (here, x markers). Regions of space that lay outside of the manifold, or on the manifold but far from any training examples, are colored white: the classifier would still assign a label to these points, but since they have not been trained the label is undetermined (and likely wrong; exactly *how likely* is what determines the classifier's generalization performance). Adversarial attacks aim to take a given point out of its colored region and onto a white region, with minimal perturbation. This can often be achieved by taking the point off the manifold (see 'attack' arrow). Adversarial training can defend against these attacks by expanding the data manifold in all directions (dashed volume). Alternatively, generative defense methods would learn to project input points onto the data manifold (green arrow). This would protect against off-manifold attacks; however, there are still gaps within the manifold (white regions) that an attacker could use to define adversarial perturbations.

3 Our Hypothesis

To design a robust vision model:

1. It is necessary, but not sufficient, that the network has a mechanism to keep its inputs on the original data manifold.

2. Given such a mechanism, additional robustness can be derived by semantic proximity.

The motivation for point 1 was explained earlier, when generative defense models were introduced (see also Figure 1): it is easy to produce adversarial perturbations that take an input off the manifold, because these are *by definition* the low-probability regions of the input space, regions in which the model has not been trained to associate inputs with labels. Projecting the image back onto the manifold would thus appear a necessary first step.

However, this step is not sufficient for adversarial robustness [23, 20] because the manifold is still a high-dimensional and often non-convex space, which the training dataset and supervised training procedure are often insufficient to properly cover, leaving 'gaps' in the category structure (Figure 1). In this context, we argue that an important limitation of standard supervised learning methods using 'one-hot' labels (or a

smoothed version of them) is that it results in a representation space where categories are arbitrarily organized, roughly equidistant from one-another. This, obviously, does not represent the physical world where a cat is more similar to a dog or a tiger than a machine gun or a tree. This is thus the motivation for point 2: we believe that additional semantic information could be used to restructure the representation space learned by the model, so that it more accurately reflects the semantic proximity between categories. This should aid in boosting the model's robustness against adversarial perturbations, especially for certain targeted attacks where the target category is semantically very different from the original label.

An important note here is that, just as Point 1 alone is insufficient to produce a fully robust model, Point 2 alone could not result in a satisfactory adversarial defense. This is because, even with a perfectly and densely structured manifold accurately representing the semantic proximity between categories, an attack could easily find a perturbation outside the manifold, where no performance guarantees exist. Thus, both parts of our hypothesis must go hand in hand.

We present our approach in testing this hypothesis in the next section.

4 Implementation: a proposal

Our hypothesis contains two parts, and consequently, its implementation requires at least two corresponding properties: a generative model for Point 1, and a way of incorporating semantic information for Point 2. While multiple choices could be satisfactory in both cases (as we have seen above for adversarial defenses based on generative models), we propose here one specific instantiation, and leave other (and possibly, better) implementations for future work.

4.1 Unsupervised generative model

For the first part of our hypothesis, we propose to use a generative network based on *predictive coding*. The predictive coding theory, proposed by Rao and Ballard [28] to explain information processing in the brain, posits that the higher areas of the cortical hierarchy make predictions about the activity of the lower regions. The errors in these predictions (termed 'residual errors') are then used to correct the activity of the higher areas in the brain. This update is carried out iteratively to attenuate the residual error. Our rationale to choose this predictive coding as our generative model is twofold. First, its global structure as a hierarchical system with feedback connections is much closer to the known cortical architecture than say, GANs, PixelCNNs or PixelRNNs. Second, it can be efficiently implemented as a stack of auto-encoders [29], with iterative refinement loops (Figure 2).

An autoencoder consists of an 'encoder', which maps the input to a feature space (also called 'latent space'), and a 'decoder', which is trained (in an unsupervised manner) to reconstruct the input from the latent space. To ensure that the autoencoder does not learn 'trivial' solutions, but truly extracts key representations of the input, the latent space can be constrained to be of lower dimensions than the input. Another (non-exclusive) possibility is to use a modified version of the autoencoder called a 'denoising autoencoder', which receives a corrupted version of the input, but is still trained to reconstruct the original, uncorrupted input. This change has been known to make autoencoders learn more robust representations compared to a standard autoencoder. Finally, additional sparsity constraints [30] can be included as a form of regularization to ensure that the model parameter fitting is not underconstrained.

In an autoencoder, the output of the decoder can be interpreted as a *prediction* of the input by the encoder. Indeed, autoencoders are trained to reduce the reconstruction loss - that is, the difference between the prediction of the input made by the encoder and the true input. This difference is exactly the 'residual error' that must be computed for predictive coding. An autoencoder (or a stack of autoencoders) optimizes the connection weights at each level using backpropagation in order to minimize this reconstruction loss. Instead (or in addition), a system based on predictive coding can modify the neuron activation values in a given layer to reduce the residual error/reconstruction loss in the immediately lower layer, in an iterative manner over several time steps.

In summary, the residual error/reconstruction loss is minimized by our model over two distinct time scales: for a given image presentation (with fixed model parameters), by updating neural activations over a predefined number of time steps; over several image presentations (i.e., a training 'mini-batch'), by updating the model parameters and, in particular, the connection weights via backpropagation (in fact, backpropagation *through time*, due to the recurrent nature of the predictive coding model).

If an adversarial image produced via an off-manifold perturbation is given as an input to this model, we initially expect a high residual error, which will be progressively reduced over successive predictive coding

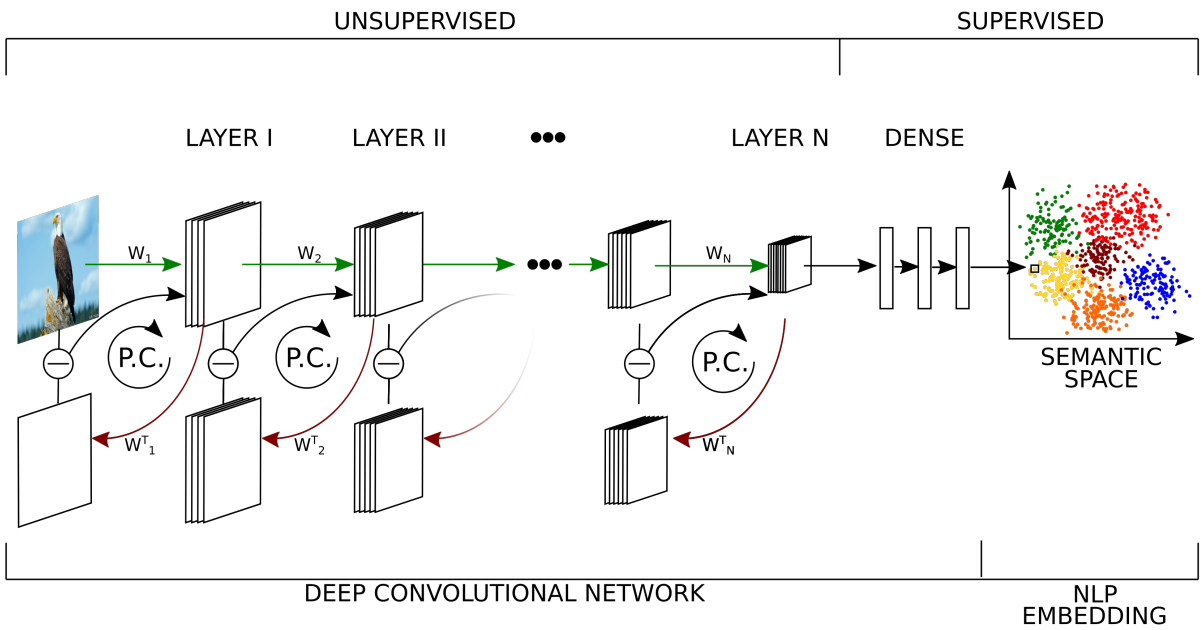


Figure 2: **The proposed model:** Layers 1 to N in the top represent the encoding layers of the convolutional autoencoders, stacked on top of one another. The layers at the bottom represent the corresponding decoding layers, which can be envisioned as *predictions* made by the higher layers. The reconstruction loss (or *residual error*) is computed between each encoding layer and the prediction from the immediately higher layer. The negative gradient of this error is then used to update activations in the higher layer. We represent this update step as a 'predictive coding' loop, denoted by P.C. in the schematic. The network weights are trained in an unsupervised manner to minimize the reconstruction loss over the entire input dataset. The last convolutional layer will be connected to a series of dense layers, and ultimately to a classification layer, trained in a supervised way. The model output, however, is not a one-hot encoded vector, but a dense vector representing the category's embedding in a NLP semantic space (e.g. Word2Vec). As in [27], we posit that backpropagation of these semantic labels into the network will modify its feature representation space in such a way that semantically related categories will move closer to one another, and unrelated categories will move further apart.

iterations, leading the model input back towards the manifold. Albeit on a simple dataset, such predictive feedback has been shown to improve the robustness of a model against adversarial examples [25].

4.2 Supervised semantic labels

The previous section explains how to design a hierarchical convolutional neural network that can extract useful features in an unsupervised manner, and behaves as a generative model, capable of reconstructing its inputs and iteratively minimizing its reconstruction error. This sets the stage for the second part of our hypothesis, in which we train this model in a supervised manner to classify images into a number of target categories, using semantic information to constrain the representation learning. For this step, we propose to use a strategy similar to the one introduced by Frome et al., known as DeViSE [27], to directly embed the output of our deep convolutional network (including a number of 'dense' layers; see Figure 2) into a semantic space defined using *Natural Language Processing* (NLP) paradigms.

Let us take the example of a Word2Vec embedding, keeping in mind that other possibilities exist (such as non-linear embeddings like BERT [31]). A multidimensional word embedding such as "Word2Vec" [32] is a neural network exposed to a very large corpus of text (e.g. Wikipedia) and trained to map any target word onto a high-dimensional vector (typically about 300 dimensions), in such a way that the vector can be used to predict the surrounding "context" words (skip-gram method). After training is completed, the resulting vector space can be used as a powerful latent representation of text data. In fact, Word2Vec or similar embeddings are often used as the first layers or pre-processing steps in NLP neural network mod-

els, such as neural machine translation systems [33]. In this space, elements cluster according to meaning, and relations between these elements can be composed using simple linear algebra. This implies that semantic proximity information of the type needed for our purposes is readily available in the distances (e.g. cosine similarity) between categories in this "semantic space". Therefore, we propose to train our deep convolutional network model in a supervised manner, not with one-hot encoded target labels as traditionally done (in the case of ImageNet, a 1000-D vector with 1 at the target position and 0 everywhere else), but with dense 300-D semantic label vectors reflecting the Word2Vec embedding of each category label. Using a similar method, Frome et al. reported that their model was capable of zero-shot learning: the output Word2Vec vector for images of novel, untrained categories tended to point toward the corresponding word in the NLP embedding. This method, however, has not yet been tested in the context of adversarial robustness.

Because semantically related categories give rise to similar Word2Vec vectors, we expect this semantic proximity information to backpropagate during supervised training and constrain the feature representations in the network. If these constraints result in semantically related categories moving closer to one another in this feature space, and semantically unrelated categories moving further apart, then it should be more difficult to design a targeted attack with a semantically unrelated adversarial target (e.g. turtle $\Longrightarrow$ rifle). This, of course, would come at the cost of increased susceptibility to confusions between semantically related items (e.g. turtle $\Longrightarrow$ lizard), but such errors (that even human observers might sometimes make) could be more acceptable for a number of AI applications.

5 Future steps

Testing the proposed model will require, at least:

- Proving that our generative model can retain inputs on (or close to) the original data manifold. This amounts to verifying that the reconstruction loss at each hierarchical level decreases both over the course of unsupervised training (by optimizing the model weights), and over successive predictive coding iterations for a given image (by optimizing neural activations in each layer).

- Ensuring that the resulting model, after supervised training with semantic labels, can perform accurate image classification. A reasonable cost in performance relative to the current state-of-the-art may be acceptable (keeping in mind that current deep learning models can perform above 78% correct on ImageNet, but their accuracy, even using state-of-the-art defenses, falls down to 0% with adversarial inputs [12]).

- Measuring adversarial robustness for a range of white-box and black-box *targeted* adversarial attacks.

As we work on these steps, it will be necessary to optimize several choices about model architecture and hyperparameters: number of layers, channels per layer, denoising and sparsity constraints, number of predictive coding iterations, greedy layer-wise vs. end-to-end training, relative weight of the unsupervised reconstruction loss vs. supervised classification loss, etc. We anticipate to have preliminary results on these experiments to report during the conference.

As future extensions of this model, we are considering the following:

1. Instead of building a full unsupervised generative model from scratch, and then making it suitable for image classification, we intend to test the possibility of adapting current state-of-the-art classification models (e.g. VGG19, InceptionV3) to our purposes. This would imply freezing the feed-forward model weights (e.g. up to the dense layers), and learning the necessary feed-back weights for successful predictive coding (i.e. manifold projection). Finally, the classification head would be replaced by a word embedding space, and the model fine-tuned for classification in this semantic space.

2. A final variant of the model could be designed to derive its semantic proximity relations directly from brain data (e.g. fMRI or EEG) rather than NLP embedding spaces. This would ensure that the neural network represents categories (and their relations) exactly as a human brain would.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical

image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, Ieee, 2009.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[4] T. Gebhart and P. Schrater, "Adversarial examples target topological holes in deep networks," *arXiv preprint arXiv:1901.09496*, 2019.

[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[6] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," *arXiv preprint arXiv:1710.10766*, 2017.

[7] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[8] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," *arXiv preprint arXiv:1606.04435*, 2016.

[9] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, and A. Madry, "On evaluating adversarial robustness," *arXiv preprint arXiv:1902.06705*, 2019.

[10] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, IEEE, 2017.

[11] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519, ACM, 2017.

[12] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[13] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," *arXiv preprint arXiv:1711.00117*, 2017.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[15] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387, IEEE, 2016.

[16] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv preprint arXiv:1712.04248*, 2017.

[17] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[18] A. Nayebi and S. Ganguli, "Biologically inspired protection of deep networks from adversarial attacks," *arXiv preprint arXiv:1703.09202*, 2017.

[19] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," *arXiv preprint arXiv:1803.01442*, 2018.

[20] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," *arXiv preprint arXiv:1805.06605*, 2018.

[21] S. Shen, G. Jin, K. Gao, and Y. Zhang, "Ape-gan: Adversarial perturbation elimination with gan," *arXiv preprint arXiv:1707.05474*, 2017.

[22] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 135–147, ACM, 2017.

[23] A. Ilyas, A. Jalal, E. Asteri, C. Daskalakis, and A. G. Dimakis, "The robust manifold defense: Adversarial training using generative models," *arXiv preprint arXiv:1712.09196*, 2017.

[24] B. Sun, N.-h. Tsai, F. Liu, R. Yu, and H. Su, "Adversarial defense by stratified convolutional sparse coding," *arXiv preprint arXiv:1812.00037*, 2018.

[25] J. Orchard and L. Castricato, "Combating adversarial inputs using a predictive-estimator network," in *International Conference on Neural Information Processing*, pp. 118–125, Springer, 2017.

[26] A. Mustafa, S. H. Khan, M. Hayat, J. Shen, and L. Shao, "Image super-resolution as a defense against adversarial attacks," *arXiv preprint arXiv:1901.01677*, 2019.

[27] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov, "Devise: A deep visual-semantic embedding model," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 2121–2129, Curran Associates, Inc., 2013.

[28] R. P. Rao and D. H. Ballard, "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects," *Nature neuroscience*, vol. 2, no. 1, p. 79, 1999.

[29] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[30] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, p. 607, 1996.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[32] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[33] P. Shapiro and K. Duh, "Morphological word embeddings for Arabic neural machine translation in low-resource settings," in *Proceedings of the Second Workshop on Subword/Character LEvel Models*, (New Orleans), pp. 1–11, Association for Computational Linguistics, June 2018.

Revisite des "random Fourier features" basée sur l'apprentissage PAC-Bayésien via des points d'intérêts

Léo Gautheron[1], Pascal Germain[2], Amaury Habrard[1], Gaël Letarte[3], Emilie Morvant[1], Marc Sebban[1], and Valentina Zantedeschi

[1]Univ Lyon, UJM-Saint-Etienne, CNRS, Institut d Optique Graduate School, Laboratoire Hubert Curien UMR 5516, Saint-Etienne, France
[2]Equipe-projet Modal, Inria Lille - Nord Europe, Villeneuve d'Ascq, France
[3]Département d'informatique et de génie logiciel, Université Laval, Québec, Canada

May 29, 2019

Abstract

Cet article résume et étend notre travail récent (Letarte et al., 2019), dans lequel nous avons revisité la méthode des *Random Fourier Features* (RFF) de Rahimi and Recht (2007) par le biais de la théorie PAC-Bayésienne. Bien que l'objectif principal des RFF soit d'approximer une fonction noyau, nous considérons ici la transformée de Fourier comme une distribution *a priori* sur un ensemble d'hypothèses trigonométriques. Cela suggère naturellement d'apprendre une distribution *a posteriori* sur cet ensemble d'hypothèses. Nous dérivons des bornes en généralisations qui sont optimisées en apprenant une distribution *pseudo-posterior* obtenue à partir d'une expression en forme close. À partir de cette études, nous proposons deux stratégies d'apprentissage basées sur des points d'intérêts : *(i)* la procédure en deux étapes proposée dans Letarte et al. (2019), où une représentation compacte des données est apprise, puis est utilisée pour apprendre un modèle linéaire, *(ii)* une nouvelle procédure, où l'on apprend en un seul temps la représentation et le modèle suivant une approche de type Boosting.

Mots-clefs: PAC-Bayes, Random Fourier Features, Landmarks, Boosting

1 Introduction

Kernel methods (Shawe-Taylor and Cristianini, 2004), such as Support Vector Machines (Boser et al., 1992; Vapnik, 1998), map data in a high dimensional space in which a linear predictor can solve the learning problem at hand. The mapping space is not directly computed and the linear predictor is represented implicitly thanks to a kernel function. This is the powerful kernel trick: the kernel function computes the scalar product between two data points in this high dimension space. However, kernel methods notoriously suffer from two drawbacks. On the first hand, computing all the scalar products for all the learning samples is costly: $O(n^2)$ for many kernel-based methods, where n is the number of training data point. On the other hand, one has to select a kernel function adapted to the learning problem for the algorithm to succeed.

The first of these drawbacks has motivated the development of approximation methods making kernel methods more scalable, such as Nyström approximation (Williams and Seeger, 2001; Drineas and Mahoney, 2005) that constructs a low-rank approximation of the Gram matrix[1] and is data dependent, or random Fourier features (RFF) (Rahimi and Recht, 2007) that approximates the kernel with random features based on the Fourier transform and is not data dependent (a comparison between the two approaches have been conducted by Yang et al., 2012). Our work is based on the latter technique.

We start from the observation that a predictor based on kernel Fourier features can be interpreted as a weighted combination of those features according to a data independent distribution defined by the Fourier transform. We introduce an original viewpoint, where this distribution is interpreted as a *prior distribution*

[1]The Gram matrix is the $n \times n$ matrix constituted by all the kernel values computed on the learning samples.

over a space of weak hypotheses—each hypothesis being a simple trigonometric function obtained by the Fourier decomposition. This suggests that one can improve the approximation by adapting this distribution in regards to the data points: we aim at learning a *posterior distribution*. By this means, our study proposes strategies to learn a representation of the data. While this representation is not as flexible and powerful than the ones that can be learned by deep neural networks (Goodfellow et al., 2016), we think that it is worthwhile to study this strategy to eventually solve the second drawback of kernel methods which currently heavily rely on the kernel choice. With this in mind, while the majority of work related to random Fourier features focuses on the study and improvement of the kernel approximation, we propose here a reinterpretation in the light of the PAC-Bayesian theory (McAllester, 1999; Catoni, 2007). We derive generalization bounds that can be straightforwardly optimized by learning a *pseudo-posterior* thanks to a closed-form expression.

The first part of this paper is based on a very recently published work (Letarte et al., 2019), where we proposed a landmark-based learning method derived from our PAC-Bayesian analysis of random Fourier features. In this method, each landmark is learned independently, and then combined into a predictor using a linear model. The second part of this paper introduces a preliminary attempt based on a boosting strategy to learn the landmarks and the final model simultaneously.

The rest of the paper is organized as follows. Section 2 recalls the random Fourier features setting. Section 3 expresses the Fourier transform as a prior leading to a PAC-Bayesian analysis and our two landmark-based approaches in Section 4. Before concluding in Section 6, Section 5 provides experiments to illustrate the potential of our work.

2 Random Fourier Features

Problem setting. Consider a classification problem where we want to learn a predictor $f : \mathbb{R}^d \to Y$, from a d-dimensional space to a discrete output space (*e.g.*, $Y = \{0, 1, \ldots, |Y|-1\}$). The learning algorithm is given a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \sim \mathcal{D}^n$ of n *i.i.d.* samples, where $\mathcal{D}$ denotes the data generating distribution over $\mathbb{R}^d \times Y$. We consider a positive-semidefinite (PSD) kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to [-1, 1]$. Kernel machines learn predictors of the form

$$f(\mathbf{x}) \;=\; \sum_{i=1}^{n} \alpha_i \, k(\mathbf{x}_i, \mathbf{x}) \,, \tag{1}$$

by optimizing the values of vector $\boldsymbol{\alpha} \in \mathbb{R}^n$.

Fourier features. When n is large, running a kernel machine algorithm (like SVM or kernel ridge regression) is expensive in memory and running time. To circumvent this problem, Rahimi and Recht (2007) introduced the *random Fourier features* as a way to approximate the value of a *shift-invariant kernel*, *i.e.*, relying on the value of $\boldsymbol{\delta} = \mathbf{x} - \mathbf{x}' \in \mathbb{R}^d$, which we write

$$k(\boldsymbol{\delta}) \;=\; k(\mathbf{x} - \mathbf{x}') \;=\; k(\mathbf{x}, \mathbf{x}')$$

interchangeably. Let the distribution $p(\boldsymbol{\omega})$ be the Fourier transform of the shift-invariant kernel k,

$$p(\boldsymbol{\omega}) \;=\; \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} k(\boldsymbol{\delta}) \, e^{-i\,\boldsymbol{\omega}\cdot\boldsymbol{\delta}} d\,\boldsymbol{\delta} \,. \tag{2}$$

Now, by writing k as the inverse of the Fourier transform p, and using trigonometric identities, we obtain:

$$\begin{aligned} k(\mathbf{x} - \mathbf{x}') &= \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) e^{i\,\boldsymbol{\omega}\cdot(\mathbf{x}-\mathbf{x}')} d\,\boldsymbol{\omega} \\ &= \mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim p} e^{i\,\boldsymbol{\omega}\cdot(\mathbf{x}-\mathbf{x}')} \\ &= \mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim p} \Big[\cos\big(\boldsymbol{\omega}\cdot(\mathbf{x}-\mathbf{x}')\big) + i \sin\big(\boldsymbol{\omega}\cdot(\mathbf{x}-\mathbf{x}')\big) \Big] \\ &= \mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim p} \cos\big(\boldsymbol{\omega}\cdot(\mathbf{x}-\mathbf{x}')\big) \,. \end{aligned} \tag{3}$$

Rahimi and Recht (2007) suggest expressing the above $\cos\big(\boldsymbol{\omega}\cdot(\mathbf{x}-\mathbf{x}')\big)$ as a product of two features. One way to achieve this is to map every input example into

$$\mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x}) \;=\; \big(\cos(\boldsymbol{\omega}\cdot\mathbf{x}),\ \sin(\boldsymbol{\omega}\cdot\mathbf{x})\big) \,. \tag{4}$$

The random variable $\mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x}) \cdot \mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x}')$, with $\boldsymbol{\omega}$ drawn from p, is an unbiased estimate of $k(\mathbf{x}-\mathbf{x}')$. Indeed, we recover from Equation (3) and Equation (4):

$$\begin{aligned} &\mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim p} \mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x}) \cdot \mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x}') \\ &\quad= \mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim p} \Big[\cos(\boldsymbol{\omega}\cdot\mathbf{x}) \cos(\boldsymbol{\omega}\cdot\mathbf{x}') + \sin(\boldsymbol{\omega}\cdot\mathbf{x}) \sin(\boldsymbol{\omega}\cdot\mathbf{x}') \Big] \\ &\quad= \mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim p} \cos\big(\boldsymbol{\omega}\cdot(\mathbf{x}-\mathbf{x}')\big) \,. \end{aligned}$$

To reduce the variance in the estimation of $k(\mathbf{x}-\mathbf{x}')$, the idea is to sample K points *i.i.d.* from p:

$\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_K$. Then, each training sample $\mathbf{x} \in \mathbb{R}^d$ is mapped to a new feature vector in $\mathbb{R}^{2K}$:

$$\boldsymbol{\phi}(\mathbf{x}) = \frac{1}{\sqrt{K}}\Big(\cos(\boldsymbol{\omega}_1 \cdot \mathbf{x}) \,,\ \ldots\ ,\ \cos(\boldsymbol{\omega}_K \cdot \mathbf{x}) \,, \quad \sin(\boldsymbol{\omega}_1 \cdot \mathbf{x}) \,,\ \ldots\ ,\ \sin(\boldsymbol{\omega}_K \cdot \mathbf{x}) \Big) \,. \tag{5}$$

Thus, we have $k(\mathbf{x}-\mathbf{x}') \approx \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')$ when K is "large enough". This provides a decomposition of the PSD kernel k that differs from the classical one (as discussed in Bach, 2017). By learning a linear predictor on the transformed training set $S \mapsto \{(\boldsymbol{\phi}(\mathbf{x}_i), y_i)\}_{i=1}^n$ through an algorithm like a linear SVM, we recover a predictor equivalent to the one learned by a kernelized algorithm. That is, we learn a weight vector $\mathbf{w} = (w_1, \ldots, w_{2K}) \in \mathbb{R}^{2K}$ and we predict the label of a sample $\mathbf{x} \in \mathbb{R}^d$ by computing, in place of Equation (1),

$$f(\mathbf{x}) \ = \ \sum_{j=1}^{2K} w_j \, \boldsymbol{\phi}_j(\mathbf{x}) \,. \tag{6}$$

3 The Fourier Transform as a Prior Distribution

As described in the previous section, the random Fourier features trick has been introduced to reduce the running time of kernel learning algorithms. Consequently, most of the subsequent work study and/or improve the properties of the kernel approximation (*e.g.*, Yu et al. (2016); Rudi and Rosasco (2017); Bach (2017); Choromanski et al. (2018)) with some notable exceptions, as the *kernel learning* algorithms of Yang et al. (2015), Sinha and Duchi (2016), and Oliva et al. (2016).

We aim at reinterpreting the Fourier transform—*i.e.*, the distribution p of Equation (2)—as a *prior distribution* over the feature space. It can be seen as an alternative representation of the prior knowledge that is encoded in the choice of a specific kernel function, that we denote k_p from now on. In accordance with Equation (3), each feature obtained from a vector $\boldsymbol{\omega} \in \mathbb{R}^d$ can be seen as a hypothesis

$$h_{\boldsymbol{\omega}}(\boldsymbol{\delta}) := \cos(\boldsymbol{\omega} \cdot \boldsymbol{\delta}) \,.$$

Henceforth, the kernel is interpreted as a predictor performing a p-weighed aggregation of weak hypotheses. This alternative interpretation of distribution p as a prior over hypotheses naturally suggests to *learn a posterior distribution* over the same hypotheses. That is, we seek a distribution q giving rise to a new kernel

$$k_q(\boldsymbol{\delta}) := \mathop{\mathbf{E}}_{\boldsymbol{\omega} \sim q} h_{\boldsymbol{\omega}}(\boldsymbol{\delta}) \,.$$

In order to assess the quality of the kernel k_q, we define a loss function based on the consideration that its output should be high when two samples share the same label, and low otherwise. Hence, we evaluate the kernel on two samples $(\mathbf{x}, y) \sim \mathcal{D}$ and $(\mathbf{x}', y') \sim \mathcal{D}$ through the linear loss

$$\ell\big(k_q(\boldsymbol{\delta}), \lambda\big) := \frac{1 - \lambda \, k_q(\boldsymbol{\delta})}{2} \,, \tag{7}$$

where $\boldsymbol{\delta} = \mathbf{x} - \mathbf{x}'$ denotes a pairwise distance and λ denotes the pairwise similarity measure:

$$\lambda \ = \ \lambda(y, y') := \begin{cases} \ \ 1 & \text{if } y = y', \\ -1 & \text{otherwise.} \end{cases}$$

Furthermore, we define the *kernel alignment* generalization loss $\mathcal{L}_\Delta(k_q)$ on a "pairwise" probability distribution Δ, defined over $\mathbb{R}^d \times [-1, 1]$ as

$$\mathcal{L}_\Delta(k_q) := \mathop{\mathbf{E}}_{(\boldsymbol{\delta}, \lambda) \sim \Delta} \ell\big(k_q(\boldsymbol{\delta}), \lambda\big) \,. \tag{8}$$

Note that any data generating distribution $\mathcal{D}$ over input-output spaces $\mathbb{R}^d \times Y$ automatically gives rise to a "pairwise" distribution $\Delta_{\mathcal{D}}$. By a slight abuse of notation, we write $\mathcal{L}_{\mathcal{D}}(k_q)$ the corresponding generalization loss, and the associated *kernel alignment* empirical loss is defined as

$$\widehat{\mathcal{L}}_S(k_q) := \frac{1}{n(n-1)} \sum_{i,j=1, i \neq j}^{n} \ell\big(k_q(\boldsymbol{\delta}_{ij}), \lambda_{ij}\big) \,, \tag{9}$$

where for a pair of examples $\{(\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j)\} \in S^2$ we have $\boldsymbol{\delta}_{ij} := (\mathbf{x}_i - \mathbf{x}_j)$ and $\lambda_{ij} := \lambda(y_i, y_j)$.

Starting from this reinterpretation of the Fourier transform, we provide in the rest of the paper one PAC-Bayesian analysis by combining n PAC-Bayesian bounds: instead of considering all the possible pairs of data points, we fix one point and we study the generalization ability for all the pairs involving it.

Note that in Letarte et al. (2019), we provide another analysis based on the fact that the loss can be expressed as a second-order U-statistics allowing us to propose a PAC-Bayesian interpretation of kernel alignment algorithm of Sinha and Duchi (2016).

4 PAC-Bayesian Analysis and Landmarks Learning

Due to the linearity of the loss function ℓ, we can rewrite the loss of k_q as the q-average loss of every

hypothesis. Indeed, Equation (8) becomes

$$\begin{aligned}\mathcal{L}_{\mathcal{D}}(k_q) &= \mathop{\mathbf{E}}_{(\boldsymbol{\delta},\lambda)\sim\Delta_{\mathcal{D}}} \ell\Big(\mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim q} h_{\boldsymbol{\omega}}(\boldsymbol{\delta}),\lambda\Big)\\ &= \mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim q}\mathop{\mathbf{E}}_{(\boldsymbol{\delta},\lambda)\sim\Delta_{\mathcal{D}}} \ell(h_{\boldsymbol{\omega}}(\boldsymbol{\delta}),\lambda)\\ &= \mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim q} \mathcal{L}_{\mathcal{D}}(h_{\boldsymbol{\omega}})\,.\end{aligned}$$

The above q-expectation of losses $\mathcal{L}_{\mathcal{D}}(h_{\boldsymbol{\omega}})$ turns out to be the quantity bounded by most PAC-Bayesian generalization theorems (sometimes referred as the *Gibbs risk* in the literature), excepted that such results usually apply to the loss over samples instead of distances. Hence, we use PAC-Bayesian bounds to obtain generalization guarantees on $\mathcal{L}_{\mathcal{D}}(k_q)$ from its empirical estimate of Equation (9), that we can rewrite as

$$\begin{aligned}\widehat{\mathcal{L}}_S(k_q) &= \frac{1}{n^2-n}\sum_{i,j=1;i\neq j}^{n} \ell\Big(\mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim q} h_{\boldsymbol{\omega}}(\boldsymbol{\delta}),\lambda_{ij}\Big)\\ &= \mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim q}\widehat{\mathcal{L}}_S(h_{\boldsymbol{\omega}})\,.\end{aligned}$$

However the *classical* PAC-Bayesian theorems cannot be applied directly to bound $\mathcal{L}_{\mathcal{D}}(k_q)$, as the empirical loss $\widehat{\mathcal{L}}_S(k_q)$ would require to be computed from *i.i.d.* observations of $\Delta_{\mathcal{D}}$. Instead, the empirical loss involves *dependent* samples, as it is computed from n^2-n pairs formed by n elements from $\mathcal{D}$.

4.1 Generalization Bound

A straightforward approach to apply *classical* PAC-Bayesian results is to bound separately the loss associated with each training sample. That is, for each $(\mathbf{x}_i, y_i) \in S$, we define

$$\mathcal{L}^i_{\mathcal{D}}(h_{\boldsymbol{\omega}}) := \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \ell\Big(h_{\boldsymbol{\omega}}(\mathbf{x}_i-\mathbf{x}),\lambda(y_i,y)\Big)\,, \quad (10)$$

$$\text{and } \widehat{\mathcal{L}}^i_S(h_{\boldsymbol{\omega}}) := \frac{1}{n-1}\sum_{j=1,j\neq i}^{n} \ell\Big(h_{\boldsymbol{\omega}}(\mathbf{x}_i-\mathbf{x}_j),\lambda(y_i,y_j)\Big)\,.$$

Thus, the next theorem gives a generalization guarantee on $\mathcal{L}^i_{\mathcal{D}}(k_q)$ relying namely on the empirical estimate $\widehat{\mathcal{L}}^i_S(k_q)$ and the Kullback-Leibler divergence $\mathrm{KL}(q\|p) = \mathbf{E}_{\boldsymbol{\omega}\sim q} \ln \frac{q(\boldsymbol{\omega})}{p(\boldsymbol{\omega})}$ between the prior p and the learned posterior q. Note that the statement of Theorem 1 is obtained straightforwardly from Alquier et al. (2016, Theorem 4.1 and Lemma 1), but can be recovered easily from Lever et al. (2013).

Theorem 1. *For $t > 0$, $i \in \{1,\dots,n\}$, and a prior distribution p over $\mathbb{R}^d$, with probability $1-\varepsilon$ over the choice of $S \sim \mathcal{D}^n$, we have for all q on $\mathbb{R}^d$:*

$$\mathcal{L}^i_{\mathcal{D}}(k_q) \le \widehat{\mathcal{L}}^i_S(k_q) + \frac{1}{t}\left(\mathrm{KL}(q\|p) + \frac{t^2}{2(n-1)} + \ln\frac{1}{\varepsilon}\right).$$

By the union bound, and using the fact that $\mathcal{L}_{\mathcal{D}}(k_q) = \mathbf{E}_{(\mathbf{x}_i,y_i)\sim\mathcal{D}} \mathcal{L}^i_{\mathcal{D}}(k_q)$, we prove the following corollary.

Corollary 2. *For $t > 0$ and a prior distribution p over $\mathbb{R}^d$, with probability $1-\varepsilon$ over the choice of $S \sim \mathcal{D}^n$, we have for all q on $\mathbb{R}^d$:*

$$\mathcal{L}_{\mathcal{D}}(k_q) \le \widehat{\mathcal{L}}_S(k_q) + \frac{2}{t}\left(\mathrm{KL}(q\|p) + \frac{t^2}{2(n-1)} + \ln\frac{n+1}{\varepsilon}\right).$$

Proof. We want to bound

$$\begin{aligned}\mathcal{L}_{\mathcal{D}}(k_q) &= \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim\mathcal{D}}\mathop{\mathbf{E}}_{(\mathbf{x}',y')\sim\mathcal{D}}\mathop{\mathbf{E}}_{\boldsymbol{\omega}\sim q} \ell\Big(h_{\boldsymbol{\omega}}(\mathbf{x}-\mathbf{x}'),\lambda(y,y')\Big)\\ &= \mathop{\mathbf{E}}_{(\mathbf{x}',y')\sim\mathcal{D}} \mathcal{L}'_{\mathcal{D}}(k_q)\,,\end{aligned}$$

where $\mathcal{L}'_{\mathcal{D}}(k_q)$ is the alignment loss of the kernel k_q centered on $(\mathbf{x}', y') \sim \mathcal{D}$ (see Equation (10)).

Let $t > 0$ and p a distribution on $\mathbb{R}^d$. By applying the PAC-Bayesian theorem, with $\varepsilon_0 \in (0,1)$, with probability at least $1-\epsilon_0$ over the random choice of $S \sim \mathcal{D}^n$, we have for all q on $\mathbb{R}^d$

$$\mathcal{L}_{\mathcal{D}}(k_q) \le \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}^i_{\mathcal{D}}(k_q) + \frac{1}{t}\left[\mathrm{KL}(q\|p) + \frac{t^2}{2n} + \ln\frac{1}{\varepsilon_0}\right].$$

Moreover, we have that for each $i \in \{1,\dots,n\}$, with a $\varepsilon_i \in (0,1)$, with probability at least $1-\epsilon_1$ over the random choice of $S \sim \mathcal{D}^n$, we have for all q on $\mathbb{R}^d$

$$\mathcal{L}^i_{\mathcal{D}}(k_q) \le \widehat{\mathcal{L}}^i_S(k_q) + \frac{1}{t}\left[\mathrm{KL}(q\|p) + \frac{t^2}{2(n-1)} + \ln\frac{1}{\varepsilon_i}\right].$$

By combining above probabilistic results with $\varepsilon_0 = \varepsilon_1 = \cdots = \varepsilon_n = \varepsilon/(n+1)$, we obtain that, with probability at least $1-\varepsilon$,

$$\begin{aligned}&\mathcal{L}_{\mathcal{D}}(k_q) = \mathop{\mathbf{E}}_{(\mathbf{x}',y')\sim\mathcal{D}} \mathcal{L}'_{\mathcal{D}}(k_q)\\ &\le \frac{1}{n}\sum_{i=1}^{n}\left[\widehat{\mathcal{L}}^i_S(k_q) + \frac{1}{t}\left[\mathrm{KL}(q\|p) + \frac{t^2}{2(n-1)} + \ln\frac{n+1}{\varepsilon}\right]\right]\\ &\quad + \frac{1}{t}\left[\mathrm{KL}(q\|p) + \frac{t^2}{2n} + \ln\frac{n+1}{\varepsilon}\right]\\ &= \widehat{\mathcal{L}}_S(k_q) + \frac{1}{t}\left[\mathrm{KL}(q\|p) + \frac{t^2}{2(n-1)} + \ln\frac{n+1}{\varepsilon}\right]\\ &\quad + \frac{1}{t}\left[\mathrm{KL}(q\|p) + \frac{t^2}{2n} + \ln\frac{n+1}{\varepsilon}\right]\\ &\le \widehat{\mathcal{L}}_S(k_q) + \frac{2}{t}\left[\mathrm{KL}(q\|p) + \frac{t^2}{2(n-1)} + \ln\frac{n+1}{\varepsilon}\right].\end{aligned}$$

□

Pseudo-Posterior. Since the above result is valid for any distribution q, one can compute the bound for any learned posterior distribution. Note that the bound promotes the minimization of a trade-off—parameterized by a constant t—between the empirical loss $\widehat{\mathcal{L}}_S(k_q)$ and the KL-divergence between the prior p and the posterior q:

$$\widehat{\mathcal{L}}_S(k_q) + \frac{2}{t}\,\mathrm{KL}(q\|p)\,.$$

It is well-known that for fixed t, p and S, the minimum bound value is obtained with the *pseudo-Bayesian* posterior q^*, such that for $\boldsymbol{\omega} \in \mathbb{R}^d$,

$$q^*(\boldsymbol{\omega}) \;=\; \frac{1}{Z}\,p(\boldsymbol{\omega})\,\exp\Big(-\tau\,\widehat{\mathcal{L}}_S(h_{\boldsymbol{\omega}})\Big)\,, \qquad (11)$$

where $\tau{:=}\frac{1}{2}t$ and Z is a normalization constant.[2]

Note also Corollary 2's bound converges to the generalization loss $\mathcal{L}_{\mathcal{D}}(k_q)$ at rate $O\Big(\sqrt{\frac{\ln n}{n}}\Big)$ for the parameter choice $t = \sqrt{n \ln n}$.

Due to the continuity of the feature space, the pseudo-posterior of Equation (11) is hard to compute. To estimate it, one may make use of Monte Carlo (*e.g.*, Dalalyan and Tsybakov (2012)) or variational Bayes methods (*e.g.*, Alquier et al. (2016)). In this work, we explore a simpler method: we work solely from a discrete probability space.

4.2 Landmarks Learning

We now propose to leverage on the fact that Theorem 1 bounds the kernel function for the distances to a single data point, instead of learning a kernel globally for every data point as in Corollary 2. We thus aim at learning a collection of kernels (which we can also interpret as similarity functions) for a subset of the training points. We call *landmarks* these training points. The aim of this approach is to learn a new representation of the input space, mapping the data-points into compact feature vectors, from which we can learn a simple predictor.

Concretely, along with the learning sample S of n examples *i.i.d.* from $\mathcal{D}$, we consider a sample of landmarks $L = \{(\mathbf{x}_l, y_l)\}_{l=1}^{n_L}$ of n_L points *i.i.d.* from $\mathcal{D}$, and a *prior* Fourier transform distribution p. For each landmark $(\mathbf{x}_l, y_l) \in L$, let sample K points from p, denoted $\boldsymbol{\Omega}^l = \{\boldsymbol{\omega}_m^l\}_{m=1}^K \sim p^K$. Then, consider a uniform distribution P on the discrete hypothesis set $\boldsymbol{\Omega}^l$, such that $P(\boldsymbol{\omega}_m^l) = \frac{1}{K}$ and $h_m^l(\boldsymbol{\delta}){:=}\cos(\boldsymbol{\omega}_m^l \cdot \boldsymbol{\delta})$. We aim at learning a set of kernels $\{\widehat{k}_{Q^l}\}_{l=1}^{n_L}$, where each $\widehat{k}_{Q^l}$ is obtained from a distinct $(\mathbf{x}_l, y_l) \in L$,

$$\widehat{k}_{Q^l}(\mathbf{x}_l{-}\mathbf{x}_i) = \sum_{m=1}^{K} Q_m^l \cos(\boldsymbol{\omega}_m^l{\cdot}(\mathbf{x}_l{-}\mathbf{x}_i))\,, \qquad (12)$$

by computing the pseudo-posterior distribution Q^l with a fixed parameter $\beta > 0$. Thus,

$$Q_m^l \;=\; \frac{1}{Z_l}\,\exp\Big(-\beta\sqrt{n}\,\widehat{\mathcal{L}}_S^l(h_m^l)\Big)\,, \qquad (13)$$

for $m{=}1,\ldots,K$; Z_l being the normalization constant. Note that Equation (13) gives the minimum of Theorem 1 with $t = \beta\sqrt{n}$. That is, $\beta = 1$ corresponds to the regime where the bound converges. Moreover, similarly to Corollary 2, generalization guarantees are obtained simultaneously for the n_L computed distributions thanks to the union bound and Theorem 1. Thus, with probability $1{-}\varepsilon$, for all $\{Q^l\}_{l=1}^{n_L}$:

$$\mathcal{L}_{\mathcal{D}}^l(\widehat{k}_{Q^l}) \le \widehat{\mathcal{L}}_S^l(\widehat{k}_{Q^l}){+}\frac{1}{t}\Big(\mathrm{KL}(Q^l\|P){+}\frac{t^2}{2(n{-}1)}{+}\ln\frac{n_L}{\varepsilon}\Big),$$

where $\mathrm{KL}(Q^l\|P) = \ln K + \sum_{j=1}^K Q_j^l \ln Q_j^l$.

Once all pseudo-posterior are computed thanks to Equation (13), our first landmarks-based approach is to map samples $\mathbf{x} \in \mathbb{R}^d$ to n_L similarity features:

$$\boldsymbol{\psi}(\mathbf{x}){:=}\Big(\widehat{k}_{Q^1}(\mathbf{x}_1{-}\mathbf{x}),\ldots,\widehat{k}_{Q^{n_L}}(\mathbf{x}_{n_L}{-}\mathbf{x})\Big), \qquad (14)$$

and to learn in a second step a linear predictor on the transformed training set. Note that, this mapping is not a kernel map anymore and is somehow similar to the mapping proposed by Balcan et al. (2008b,a); Zantedeschi et al. (2018) for a similarity function that is more general than a kernel but fixed for each landmark.

4.3 Boosting the Landmarks

In this subsection we present an exploratory work in order to propose *in fine* an algorithm allowing to learn the representation and the predictor simultaneously. As pointed out in Section 3, a kernel can be interpreted as an aggregation of weak hypotheses. Thus, a natural solution is to learn a majority vote of weak hypotheses through a Boosting-based algorithm.

[2] This trade-off is the same one involved in some other PAC-Bayesian bounds for *i.i.d.* data (*e.g.*, Catoni (2007)). As discussed in Zhang (2006); Grünwald (2012); Germain et al. (2016), there is a similarity between the minimization of such PAC-Bayes bounds and the Bayes update rule.

We follow the well-known Adaboost algorithm (Freund and Schapire, 1997) with confidence-rated predictions, since our weak hypotheses return a value in $[-1, 1]$. Basically, Adaboost is an iterative algorithm, where at each iteration it selects/learns a weak predictor to add to the majority vote (with a certain weight). Then, the learning sample is reweighted in order to give more importance to misclassified examples in the next iteration.

Algorithm 1 presents the Adaboost algorithm instantiated to our context; we give more details in the following. First of all, we denote by D_i^t the weight

Algorithm 1: AdaBoost for landmark-based random features with Confidence-rated Predictions

Input : Learning set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
Landmark set $L = \{(\mathbf{x}_l, y_l)\}_{l=1}^{n_L}$,
Number of iterations T.

Output: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha^t \widehat{k}_{Q^t}(\mathbf{x}_t - \mathbf{x})\right)$

0. $D_i^t \leftarrow \frac{1}{n}$ for $i = \{1, \ldots, n\}$

for $t = 1, \ldots, T$ **do**

1. Draw $\mathbf{\Omega}^t \sim p^K$

2. $(\mathbf{x}_t, y_t, Q^t) \leftarrow \underset{\substack{(\mathbf{x},y)\in L,\\ Q\in\mathbb{R}^K}}{\text{argmax}} \sum_{i=1}^{n} D_i^t y_i \widehat{k}_Q(\mathbf{x} - \mathbf{x}_i)$

3. $\alpha^t \leftarrow \frac{1}{2} \ln \frac{1+r^t}{1-r^t}$,
with $r^t = \sum_{i=1}^{n} D_i^t y_i \widehat{k}_{Q^t}(\mathbf{x}_t - \mathbf{x}_i)$

4. $D_i^{t+1} \leftarrow D_i^t \exp(-\alpha^t \widehat{k}_{Q^t}(\mathbf{x}_t - \mathbf{x}_i) y_i)$,
for $i = \{1, \ldots, n\}$

5. $D_i^{t+1} \leftarrow \frac{D_i^{t+1}}{\sum_{i=1}^n D_i^{t+1}}$, for $i = \{1, \ldots, n\}$

end for

associated to an example $(\mathbf{x}_i, y_i) \in S$ at iteration t, such that $\sum_{i=1}^n D_i^t = 1$; at the first iteration $\forall i \in \{1, \ldots, n\}$, $D_i^1 = \frac{1}{n}$ (line **0** of the algorithm). At a given iteration t, we sample K points from the *prior* Fourier distribution p, denoted by $\mathbf{\Omega}^t = \{\boldsymbol{\omega}_m^t\}_{m=1}^K \sim p^K$ (line **1**). Then, in line **2** we select the optimal landmark $(\mathbf{x}_t, y_t)$ from L and the associated pseudo-posterior Q^t distribution such that[3]

$$\underset{\substack{(\mathbf{x},y)\in L,\\ Q\in\mathbb{R}^K}}{\text{argmax}} \sum_{i=1}^{n} D_i^t y_i \underbrace{\left[y \sum_{m=1}^{K} Q_m \cos(\boldsymbol{\omega}_m^t \cdot (\mathbf{x} - \mathbf{x}_i)) \right]}_{\widehat{k}_Q(\mathbf{x} - \mathbf{x}_i)} . \quad (15)$$

That is, Q^t is the distribution over $\boldsymbol{\omega}^t$ that minimizes the D^t-weighted loss (Z_t is a normalization constant):

$$Q_m^t = \frac{1}{Z_t} \exp\Big(-\beta\sqrt{n}\, \widehat{\mathcal{L}}_S^t(h_m^t) \Big),$$
$$\text{where, } \widehat{\mathcal{L}}_S^t(h_m^t) = \sum_{i=1}^{n} D_i^t\, \ell\Big(h_m^t(\mathbf{x}_t, \mathbf{x}_i), \lambda(y_t, y_i)\Big).$$

Note that $\sum_{i=1}^n D_i^t y_i \widehat{k}_{Q^t}(\mathbf{x}_t - \mathbf{x}_i)$ is in fact the usual measure of correlation between the labels and the predictions in Adaboost. Thus, this step coincides to the classical Adaboost step consisting in learning a weak predictor. Lines **3** to **5** correspond to the classical steps in Adaboost consisting in computing the weight α^t given to the current weak predictor—here $\widehat{k}_{Q^t}(\mathbf{x}_t - \mathbf{x}_i)$—in the majority vote, and computing the reweighting of the learning examples in order to focus on misclassified examples.

This approach has the following two clear advantages regarding the two-step method, where one first learns the mapping (given a set of landmarks), secondly one needs to learn the final predictor.

- Adaboost allows to construct iteratively the mapping by selecting one landmark—the one that appears to currently be the best—at each step.
- The final predictor is learned at the same time, and the learning procedure can be stopped when the empirical loss stops decreasing.

Consequently, the final mapping does not necessarily needs all the landmark points, and the ones selected will be more suitable for the considered task.

5 Experiments

We conduct experiments to compare our new boosting-based method with our two-step procedure. Let us

[3] Contrary to Equation (12), we include the landmark label y in the kernel expression to enforce the predictor to be accurate on its center (*i.e.*, $(\widehat{k}_Q(\mathbf{x} - \mathbf{x}) = y)$. It was not mandatory for the method presented in Section 4.2, as far as the linear predictor learned on the mapping of Equation (14) is allowed to have both positive and negative weights.

Dataset	Boosting methods				PB
	RBF	LinearSVM	Tree	PB	
wine	96.3 ± 1.3	97.5 ± 2.0	94.2 ± 2.2	**97.6** ± 1.8	96.1 ± 2.0
sonar	71.1 ± 3.7	73.4 ± 4.6	76.3 ± 3.2	**78.8** ± 3.4	70.6 ± 4.1
glass	69.2 ± 3.3	72.9 ± 4.6	**79.6** ± 4.5	75.6 ± 3.6	75.5 ± 3.9
newthyroid	90.2 ± 1.6	87.5 ± 3.3	91.5 ± 3.5	94.7 ± 1.5	**95.3** ± 1.9
heart	**83.5** ± 1.4	81.5 ± 2.5	78.0 ± 2.8	83.5 ± 1.7	80.3 ± 2.5
bupa	59.8 ± 3.7	65.0 ± 3.6	65.6 ± 4.6	**66.3** ± 3.1	64.3 ± 3.9
iono	80.2 ± 4.3	87.3 ± 1.8	89.7 ± 2.4	86.6 ± 1.3	**91.1** ± 1.8
wdbc	95.3 ± 1.0	95.6 ± 1.0	95.2 ± 1.4	**96.6** ± 0.9	95.3 ± 1.3
balance	94.3 ± 1.0	94.1 ± 2.0	95.0 ± 1.9	**95.7** ± 1.4	94.5 ± 2.1
australian	82.8 ± 1.0	**86.1** ± 1.3	84.2 ± 1.6	84.9 ± 1.5	85.0 ± 1.3
pima	75.5 ± 1.9	**75.8** ± 2.0	74.3 ± 1.3	75.4 ± 1.7	74.8 ± 1.7
german	70.2 ± 0.2	71.8 ± 2.5	71.5 ± 1.5	**72.1** ± 1.6	71.0 ± 1.4
splice	68.9 ± 2.0	83.3 ± 0.8	**96.1** ± 0.4	85.6 ± 0.8	84.3 ± 0.9
spambase	80.8 ± 0.9	91.6 ± 0.6	**94.3** ± 0.3	91.5 ± 0.3	91.3 ± 0.6
Mean	79.9 ± 1.9	83.1 ± 2.3	**84.7** ± 2.3	84.6 ± 1.8	83.5 ± 2.1
Mean Rank	4.20	2.93	2.80	**1.87**	3.20

Table 1: Accuracy on the test set averaged on 20 random splits.

recall that our new method is a first attempt at simultaneously learning the classifier and the representation.

We will refer to our boosting-based method as Boosting PB and to our two-step procedure as PB.

For Boosting PB, we consider weak classifiers as described in Equation (15) and tune the parameter $\beta \in 10^{\{-1,\dots,5\}}$.

For PB, for every landmark of L, we learn a *similarity measure*, and thus the posterior distributions associated to each landmark, following Equation (13) minimizing the PAC-Bayesian bound. Then, we learn a linear SVM on the mapped training set obtained by Equation (14). We select by cross-validation the parameters $\beta \in 10^{\{-3,\dots,3\}}$ and $C \in 10^{\{-3,\dots,3\}}$.

Additionally, we compare our results to those of three baselines based on AdaBoost:

Boosting RBF, where we boost weak classifiers of the form of a RBF kernel between any point and a landmark:

$$k_\sigma(\mathbf{x}_l, \mathbf{x}, s) = s \exp\big(-\gamma \|\mathbf{x}_l - \mathbf{x}\|^2\big),$$

where $s \in \{-1, 1\}$. For each landmark used, we consider the kernel multiplied by a value s equal to -1 and 1 to allow the weak classifier to take negative values. For this baseline, we select by cross-validation the parameter $\gamma \in 2^{\{-5,\dots,5\}}$;

Boosting LinearSVM, where a weak classifier is a linear SVM that predicts binary labels. We plug it into the original AdaBoost algorithm (without rated predictions), and we select by cross-validation the SVM hyper-parameter $C \in 10^{\{-3,\dots,3\}}$;

Boosting Tree, where a weak classifier takes the form of a decision tree that predicts binary labels. For this method, we select by cross-validation the maximum depth of the trees in $\{1, \dots, 8\}$ and make use of the original AdaBoost algorithm, without rated predictions.

For the three methods using landmarks, the set of landmarks considered is selected randomly from 10% of the training points. For the two methods using random features, we fix the number of random features K to 100 and for a dataset having d features, the random features are drawn from $p_\sigma(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \mathbf{0}, \frac{1}{\sigma^2}\mathbf{I})$ with $\sigma^2 = \frac{1}{\sqrt{d}}$. The four methods based on boosting are run for 100 iterations.

Here, we consider 14 binary classification datasets coming mainly from the UCI repository. All datasets are normalized such that each feature has a mean of 0 and a variance of 1. For each dataset, we generate randomly 20 splits of 30% training examples and 70% test data. The hyper-parameters of all the methods are tuned by a 5-fold cross-validation on the training set. We report the mean results over the 20 splits in Table 1.

First of all, except on 3 out of 14 datasets, the use of the one-step procedure (Boosting PB) allows to im-

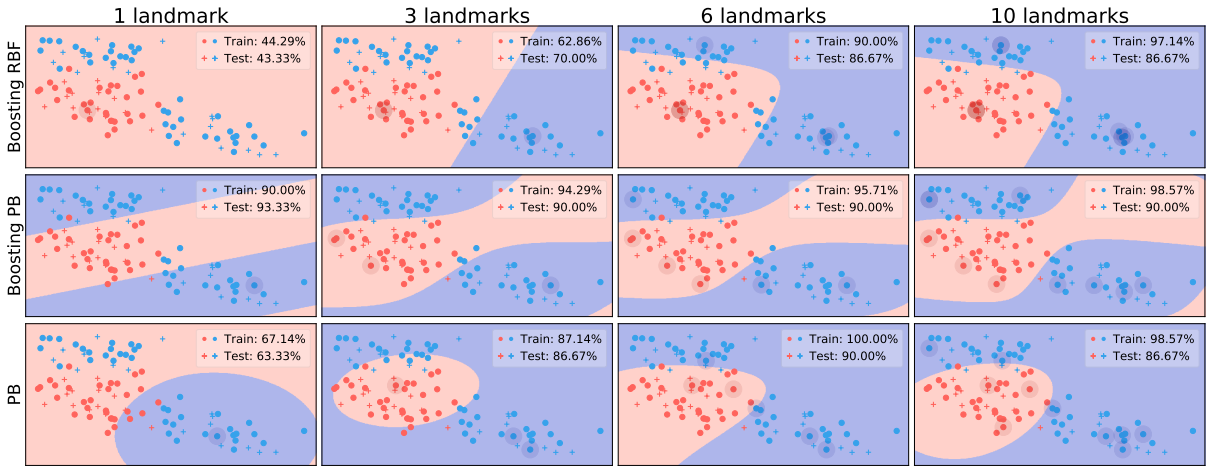


Figure 1: First, second and third rows show the decision boundary and the landmarks learned by respectively Boosting RBF, Boosting PB and PB. Each column shows the decision boundary obtained using a certain number of landmarks on the training points. Note that, the accuracy on both training and test set are given on the top right of each sub-figure.

prove the results of the two-steps procedure (PB). This is an expected result: intuitively, a one-step procedure has the power of learning a representation suitable for the classification model considered. Moreover, learning the kernel in a boosted manner leads to a significant improvement in terms of accuracy compared to deploying a pre-defined kernel, such as the RBF kernel. Finally, Boosting PB compares favorably with a rank of 1.87 to the two more traditional Adaboost approaches based on linear SVMs (rank 2.93) and decision trees (rank 2.80). Our approach is then more flexible in the sense that it has probably the capacity to capture more information about the data than the classical RBF kernel, or linear SVMs, or decision trees.

Toy experiment. To illustrate the differences between the three landmark-based methods Boosting RBF, Boosting PB and PB, we carry out an experiment on a 2-dimensional toy dataset, constructed by drawing 100 examples from 7 isotropic Gaussian clusters, each cluster belonging to one of the two classes. Figure 1 shows the decision boundary obtained by each method when trained with an increasing number of landmarks. Here, we select by cross-validation the parameters of each method. We see that Boosting RBF tends to learn simpler models and to select landmarks from the center of the clusters. On the contrary, because our methods do not make use of a fixed kernel function, they are able to better fit the problem, even with a limited number of landmarks. However, as PB selects randomly the landmarks from the training set, it needs a greater amount of landmarks than Boosting PB for fitting correctly the points.

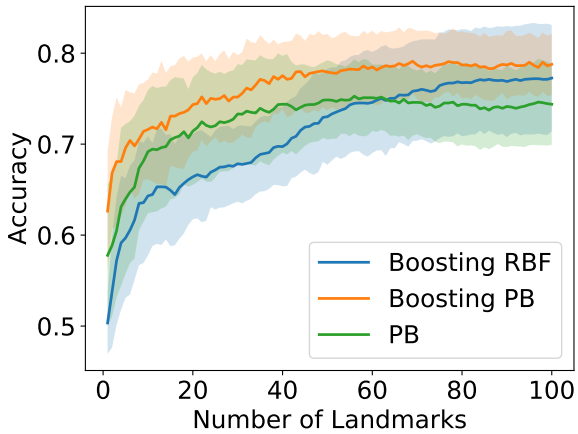


Figure 2: Mean accuracy ± standard deviation over 20 train/test splits on the "sonar" dataset as a function of the number of landmarks used to train the three methods Boosting RBF, Boosting PB and PB.

Influence of the number of landmarks. In Figure 2, we analyze the accuracy of our landmark-based methods Boosting PB and PB as a function of the

number of landmarks used to train the model. We also report the performance of Boosting RBF. We consider 20 train/test splits on the "sonar" dataset. Overall, the larger the quantity of landmarks, the better the performances for all methods. However, Boosting PB reaches good accuracy even with an amount of landmarks considerably smaller than the one required by the other methods. Notice that, at around 60 landmarks used, the performances of PB stop increasing, while Boosting RBF becomes better. This may be explained by the fact that, contrary to the boosting-based approaches that learn the landmarks at the same time than the classification model –, the random selection of landmarks in PB is not able to capture additional information for the linear SVM learned at end. With 100 landmarks, Boosting PB has the smallest standard deviation (of 3.4%), followed by PB (4.5%) and by Boosting RBF (5.8%) which suggests that learning the landmarks allows to have a more stable model on this problem w.r.t. selecting the landmarks randomly.

6 Conclusion

We elaborated an original viewpoint of the random Fourier features, proposed by Rahimi and Recht (2007) to approximate a kernel. By looking at the Fourier transform as a prior distribution over trigonometric functions, we present a generalization theorem that bounds a kernel alignment loss. Based on classical first-order PAC-Bayesian results, we derived two landmarks-based strategies that learn a compact representation of the data. While the first one consists only on constructing the representation (and then one can learn, in a second step, a predictor in this representation), the second one learns at the same time the representation and the final predictor, following a boosting-based approach.

Our current guarantees hold solely for the kernel alignment loss, and not for the predictor trained with this kernel. An important research direction is to extend the guarantees to the final predictor, which could in turn be the bedrock of a new one-step learning procedure (in the vein of Yang et al. (2015); Oliva et al. (2016)). Other research directions include the study of the RKHS associated with the learned kernel, and the extension of our study to wavelet transforms (Mallat, 2008). Furthermore, considering the Fourier transform of a kernel as a (pseudo-)Bayesian prior might lead to other original contributions. Among them, we foresee new perspectives on representation and metric learning, namely for unsupervised learning. Moreover, the promising empirical results suggest that we should continue in the direction of exploring boosting-based techniques to jointly learn the representation and the model. An interesting line of research would be to extend our work to gradient boosting Friedman (2001) and to optimize the coordinates of the landmarks instead of selecting them from the training set. Finally, we believe that learning random feature-based representation along with "good" landmarks for a given task can be helpful for other (semi-)supervised classification tasks, such as learning with imbalanced data or transfer learning.

Acknowledgments. Pascal Germain wants to thank Francis Bach for insightful preliminary discussions. This work was supported in part by the French Project APRIORI ANR-18-CE23-0015 and in part by NSERC. This research was enabled in part by support provided by Compute Canada (`www.computecanada.ca`).

References

Pierre Alquier, James Ridgway, and Nicolas Chopin. On the properties of variational approximations of gibbs posteriors. *Journal of Machine Learning Research*, 17, 2016.

Francis R. Bach. On the equivalence between kernel quadrature rules and random feature expansions. *Journal of Machine Learning Research*, 18, 2017.

Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. Improved guarantees for learning via similarity functions. In *COLT*, 2008a.

Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. A theory of learning with similarity functions. *Machine Learning*, 72(1-2):89–112, 2008b.

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, 1992.

Olivier Catoni. *PAC-Bayesian supervised classification: the thermodynamics of statistical learning*, volume 56. Inst. of Mathematical Statistic, 2007.

Krzysztof Choromanski, Mark Rowland, Tamás Sarlós, Vikas Sindhwani, Richard E. Turner, and Adrian Weller. The geometry of random features. In *AISTATS*, 2018.

Arnak S. Dalalyan and Alexandre B. Tsybakov. Sparse regression learning by aggregation and langevin monte-carlo. *J. Comput. Syst. Sci.*, 78(5), 2012.

Petros Drineas and Michael W Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(Dec), 2005.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Pascal Germain, Francis R. Bach, Alexandre Lacoste, and Simon Lacoste-Julien. PAC-Bayesian theory meets Bayesian inference. In *NIPS*, 2016.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Peter Grünwald. The safe Bayesian - learning the learning rate via the mixability gap. In *ALT*, 2012.

Gaël Letarte, Pascal Germain, and Emilie Morvant. Pseudo-bayesian learning with kernel fourier transform as prior. In *AISTATS*, 2019.

Guy Lever, François Laviolette, and John Shawe-Taylor. Tighter PAC-Bayes bounds through distribution-dependent priors. *Theor. Comput. Sci.*, 473, 2013.

Stéphane Mallat. *A Wavelet Tour of Signal Processing, 3rd Edition*. Academic Press, 2008.

David McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37(3), 1999.

Junier B Oliva, Avinava Dubey, Andrew G Wilson, Barnabás Póczos, Jeff Schneider, and Eric P Xing. Bayesian nonparametric kernel-learning. In *AISTATS*, 2016.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In *NIPS*, 2017.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

Aman Sinha and John C. Duchi. Learning kernels with random features. In *NIPS*, 2016.

Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998.

Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *NIPS*. MIT Press, 2001.

Tianbao Yang, Yu-feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *NIPS*. Curran Associates, Inc., 2012.

Zichao Yang, Andrew Gordon Wilson, Alexander J. Smola, and Le Song. A la carte - learning fast kernels. In *AISTATS*, 2015.

Felix X. Yu, Ananda Theertha Suresh, Krzysztof Marcin Choromanski, Daniel N. Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In *NIPS*, 2016.

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. Fast and provably effective multi-view classification with landmark-based svm. In *ECML-PKDD*, 2018.

Tong Zhang. Information-theoretic upper and lower bounds for statistical estimation. *IEEE Trans. Information Theory*, 52(4), 2006.

Construction d'espace latent pour la détection d'anomalies par apprentissage adversarial*

Elies Gherbi[1,2], Blaise Hanczar[1], Jean-Christophe Janodet[1] et Witold Klaudel[3]

[1]IBISC, Univ Evry, Université Paris-Saclay, 91025 Evry, France
[2]IRT SystemX, 8 avenue de la Vauve, 91120-Palaiseau
[3]Renault, 1 avenue du Golf 78288–Guyancourt

Résumé

La détection d'anomalies est un problème récurrent en *Machine Learning*. Des techniques récentes cherchent à exploiter le potentiel des GAN (*Generative Adversarial Networks*) pour détecter les anomalies de façon indirecte, dans l'espace des données ; elles se fondent sur l'idée que le générateur ne peut pas reconstruire une anomalie. Nous développons une approche alternative, basée sur un *Encoding Adversarial Network* (ANOEAN), qui projette les données dans un espace latent, où la détection d'anomalies se fait de façon directe, en calculant un score. Notre encodeur est appris par *adversarial learning*, en utilisant deux fonctions de perte, la première contraignant l'encodeur à modéliser les données normales dans un espace qui suit une distribution gaussienne, et la seconde, à projeter des données anormales en dehors de cette distribution. Nous conduisons une série d'expériences sur plusieurs bases standard, et montrons que notre approche dépasse l'état de l'art lorsqu'on utilise 10% d'anomalies lors de l'apprentissage.

Mots-clef : Détection d'anomalies, *Adversarial Learning*, GAN, EAN.

1 Introduction

La détection des anomalies a longtemps été une question de grand intérêt dans plusieurs applications [4, 9, 3]. On l'utilise dans des domaines comme la détection de fraudes, la cybersécurité, la vidéo-surveillance, la maintenance prédictive, *etc.* Les développements récents dans le *deep learning* ont permis de revisiter ce problème. Ainsi, selon la taxonomie de [8], il existe deux grandes familles de méthodes.

La première catégorie, *Representation learning*, consiste à apprendre une représentation des données normales qu'on utilise pour déclencher une alarme lorsqu'un comportement déviant est repéré. De nombreuses techniques d'apprentissage automatique ont été utilisées pour construire de tels modèles, dont les machines à vecteurs de support à une classe (OCSVM) [15] ; il s'agit dans ce cas de construire une boule (*cluster*) de faible rayon autour des données normales, et de décréter que toute anomalie est en dehors.

La deuxième catégorie, *Reconstruction based anomalie score*, part de l'hypothèse que les anomalies possèdent des caractéristiques différentes de celles des données normales. En conséquence, il est difficile de reconstruire une anomalie en utilisant un modèle appris pour la classe normale. Les *autoencoders* ont été utilisés en premier dans cette catégorie [18, 19], mais les techniques relevant du *adversarial learning* (GAN, BiGAN) sont de plus en plus populaires, plusieurs travaux ont adapté les réseaux génératifs adversariaux (GAN) pour la detection d'anomalies[14, 21, 20, 2, 8, 13, 6].

Un GAN (*Generative Adversarial Network*) est composé de deux réseaux : un générateur G, transformant les vecteurs z d'un espace latent en données artificielles, et un discriminateur D dont le rôle est de différencier les données artificielles des données réelles. On entraîne un tel modèle de façon concurrente : le générateur doit leurrer le discrimateur en apprenant à approximer la distribution des données réelles.

Dans le contexte de la détection d'anomalies, ANOGAN [14] et ses variantes [6], apprennent un GAN à partir de données normales, puis pour un exemple x de test, l'algorithme recherche un point z dans l'espace latent tel que $G(z) \approx x$; si on n'échoue à trouver un tel z,

*This work has been partially carried out in IRT SystemX, and therefore granted with public funds within the scope of the French program Investissements d'avenir. This work is part of Project "Cybersecurity for Intelligent Transport".

alors on déclare que l'exemple x est une anomalie. En d'autres termes, on cherche à calculer $z = G^{-1}(x)$ en partant du principe que G n'est inversible que pour les exemples normaux. Pour celà, on introduit une fonction de perte en reconstruction (*reconstruction loss*), typiquement $\mathcal{L}_r = \|x - G(z)\|$, qu'on cherche à minimiser en fonction de z. Cette technique est très coûteuse en temps car pour chaque exemple de test, on doit faire une descente de gradient.

Aussi, d'autres algorithmes comme EGBAD et ALAD [20, 21] se basent sur des BiGAN plutôt que des GAN, c'est-à-dire des architectures composées de trois réseaux : un générateur G, un discriminateur D ainsi qu'un encodeur E, dont le rôle est de projeter les données vers l'espace latent avec $E = G^{-1}$. Dans ce cas, chercher z tel que $G(z) \approx x$ est immédiat : il suffit de prendre $z = E(x)$. Mais en pratique, le générateur et l'encodeur sont rarement inverses l'un de l'autre [5], ceci peut conduire à des scores de reconstruction élevés pour des données normales.

Dans cet article, nous proposons une technique alternative, appelée AnoEAN pour *Anomaly detection with Encoding Adversarial Network*. L'idée est de travailler directement dans un espace latent et l'utiliser comme espace de décision.Pour se faire, nous entraînons un encodeur, par *adversarial learning*, à projeter les données normales dans une région restreinte de l'espace latent, et les anomalies en dehors de cette région. Nous supposons disposer d'une grande quantité de données normales, et d'un petit nombre d'anomalies. C'est un cadre de travail fréquent en détection d'anomalies. Ce faisant, nous éliminons les problèmes liés à l'estimation de la fonction de perte en reconstruction dont souffrent les GAN et BiGAN. Nous identifions les anomalies directement dans l'espace latent, à l'aide d'une distance de Mahalanobis calculée sur la distribution induite par les exemples normaux. Nous conduisons une série d'expériences prouvant qu'AnoEAN donne de meilleurs résultats que les techniques classiques de détection d'anomalies, notamment celles fondées sur les GAN, en utilisant à la fois la base MNIST de chiffres manuscrits [10] et deux bases de détection d'intrusions dans des réseaux (KDD'99, NSL-KDD) [12, 7].

2 Algorithme AnoEAN

Nous considérons un problème dans lequel nous surveillons les données $x \in \mathbf{R}^p$ décrivant l'état d'un système avec p variables. Notre objectif est de déclencher une alerte lorsque les données montrent que le système sort de son comportement normal. Pour

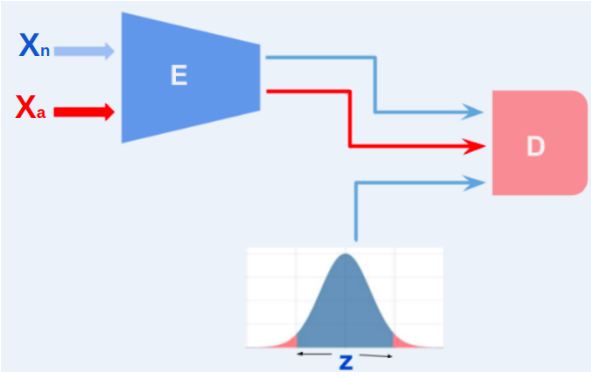


FIGURE 1 – Architecture de AnoEAN.

cela, nous construisons un score d'anomalie $a(x) : \mathbf{R}^p \to [0, +\infty[$ mesurant le degré d'anomalie d'une donnée x, les données normales devant avoir un score proche de 0. Pour apprendre cette fonction, nous disposons d'un ensemble d'apprentissage contenant N données correspondant à un comportement normal $\{x_{n_i} \mid i = 1..N\}$ et M données correspondant à des anomalies $\{x_{a_i} \mid i = 1..M\}$ avec $M \ll N$. Nous notons P_{x_n} (resp. P_{x_a}) la distribution des données normales (resp. anomalies). A noter que contrairement aux données normales, les anomalies ne forment pas une classe homogène.

Notre méthode, nommée AnoEAN pour *Anormaly detection by Encoding Adversarial Network*, consiste à projeter les données x_n et x_a dans un espace de petite dimension, appelé espace de décision, dans lequel la distribution des données normales est gaussienne. Nous appelons *encodeur*, le réseau de neurones qui projette les données de l'espace d'origine dans l'espace de décision $E(x) : \mathbf{R}^p \to \mathbf{R}^d$ avec $d \ll p$. Soit p_z une distribution gaussienne $\mathcal{N}(0, I)$ dans l'espace de décision. L'objectif de l'encodeur est de projeter les donnnées normales dans p_z et les anomalies en dehors de p_z.

Pour cela, de la même manière que les GAN, nous utilisons un deuxième réseau appelé *discriminateur* qui reçoit en entrée un vecteur de l'espace de décision et prédit si ce vecteur provient de p_z ou de l'encodeur. L'encodeur doit donc à la fois tromper le discriminateur sur la projection de données normales pour lui faire croire qu'elle proviennent de p_z et aider le discriminateur à différencier p_z de la projection des anomalies. L'architecture de notre modèle AnoEAN est représentée dans la Fig. 1. Le discriminateur et l'encodeur doivent respectivement minimiser les fonctions de coût L_D et L_E suivantes :

$$\begin{aligned} L_D = & -\mathbf{E}_{z\sim p_z}[\log(D(z))] \\ & -\mathbf{E}_{x_n\sim p_{x_n}}[\log(1-D(E(x_n)))] \\ & -\mathbf{E}_{x_a\sim p_{x_a}}[\log(1-D(E(x_a)))] \end{aligned}$$

$$\begin{aligned} L_E = & -\mathbf{E}_{x_n\sim p_{x_n}}[\log(D(E(x_n)))] \\ & -\mathbf{E}_{x_a\sim p_{x_a}}[\log(1-D(E(x_a)))] \end{aligned}$$

L'apprentissage de ce modèle suit la procédure suivante. On prend aléatoirement des exemples normaux x_n et des anomalies x_a de la base d'apprentissage et on les projette dans l'espace de décision avec l'encodeur (*i.e.*, on calcule chaque $E(x_a)$ et $E(x_n)$). On ajoute à ces exemples projetés des vecteurs aléatoires tirés de la distribution p_z afin de former un *batch* que l'on présente à l'entrée du discriminateur. Le discriminateur est modifié par descente du gradient afin de minimiser L_D ; l'encodeur est gelé pendant cette étape. Ensuite c'est au tour de l'encodeur d'être modifié afin de minimiser L_E, le discriminateur étant gelé pendant cette étape. Cette procédure est itérée jusqu'à convergence du modèle.

Une fois l'apprentissage du modèle terminé, la prédiction des anomalies ne nécessite que l'utilisation de l'encodeur. Le score d'anomalie d'un exemple correspond à la distance entre sa projection dans l'espace de décision $E(x)$ et la distribution de la projection des exemples normaux. Cette distribution est censée tendre vers $p_z = \mathcal{N}(0, I)$ au cours de l'apprentissage du modèle. Nous pourrions donc utiliser la norme de $E(x)$ comme score d'anomalie. Cependant, nos expérimentations ont montré que la distribution de la projection des exemples normaux pouvait diverger légèrement de p_z. Aussi, à la fin de l'apprentissage, nous représentons cette distribution par une distribution gausienne $\mathcal{N}(\mu, \Sigma)$ dont nous estimons les paramètres sur une base de validation. Le score d'anomalie est finalement la distance de Mahalanobis entre $E(x)$ et μ :

$$a(x) = \sqrt{(E(x)-\mu)^T \Sigma^{-1} (E(x)-\mu)}$$

3 Expérimentations et résultats

Nous présentons dans cette partie les expérimentations que nous avons conduites et leurs résultats montrant l'efficacité de notre méthode comparée à l'état de l'art.

3.1 Protocole expérimental

Dans nos expérimentations nous avons utilisé trois jeux de données : MNIST, KDD99 et NSL-KDD. MNIST [10] est une base de chiffres manuscrits qui est courament utilisée pour les problèmes de classification d'images. Bien que MNIST ne contient pas à l'origine de classe normale et d'anomalies, elle est souvent utilisée en détection d'anomalies, afin d'analyser le comportement des algorithmes et de visualiser leurs prédictions. Dans nos expérimentations, nous avons utilisé ce jeu de données de deux façons :
1) **1 contre tous** : nous considérons qu'un certain chiffre représente la classe normale et les 9 restant sont des anomalies. Pour les méthodes OCSVM, AnoGAN, ALAD, EGBAD, l'ensemble d'apprentissage est constitué de 5000 données normales ; l'ensemble de test comprend 1700 exemples dont 80% de données normales et 20% d'anomalies choisies aléatoirement parmi les 9 autres classes. Pour AnoEAN et SVM, nous injectons en plus 10% d'anomalies dans l'ensemble d'apprentissage ; celles-ci sont choisies parmi 4 classes sur 9, de sorte que certains chiffres (anomalies) n'apparaissent pas pendant l'apprentissage.
2) **n contre m** : on se base sur le même principe que le "1 contre tous", la seule différence étant que la classe normale est composée de plusieurs chiffres. Nous regroupons n classes de chiffres en une seule classe normale et traitons les m chiffres restants comme des exemples anormaux. Nous utilisons les mêmes répartitions d'exemples dans les ensembles d'apprentissage et de tests que précédemment. L'objectif est d'analyser le comportement des algorithmes dans le cas où la classe normale est hétérogène et peut se décomposer en plusieurs sous-classes.

Les deux autres jeux de données utilisés, KDD99 [12] et NSL-KDD [7], sont très courament utilisés pour évaluer les performances des algorithmes de détection d'anomalies, l'objectif étant de détecter des intrusions dans des réseaux surveillés. Nous utilisons les mêmes répartitions d'exemples dans les ensembles d'apprentissage et de tests que précédemment.

Nous comparons notre méthode avec deux approches classiques et trois méthodes basées sur les GAN.

OCSVM. Les *one-class* SVM permettent de construire un cluster à partir de données normales [15]. Nous utilisons un noyau RBF. Nous les avons également essayés avec quelques anomalies et une marge douce de 10%, mais cela n'aporte pas d'amélioration aux résultats.

SVM. Les SVMs sont sensibles aux ensembles de données déséquilibrés [16, 17, 1]. Donc de façon clas-

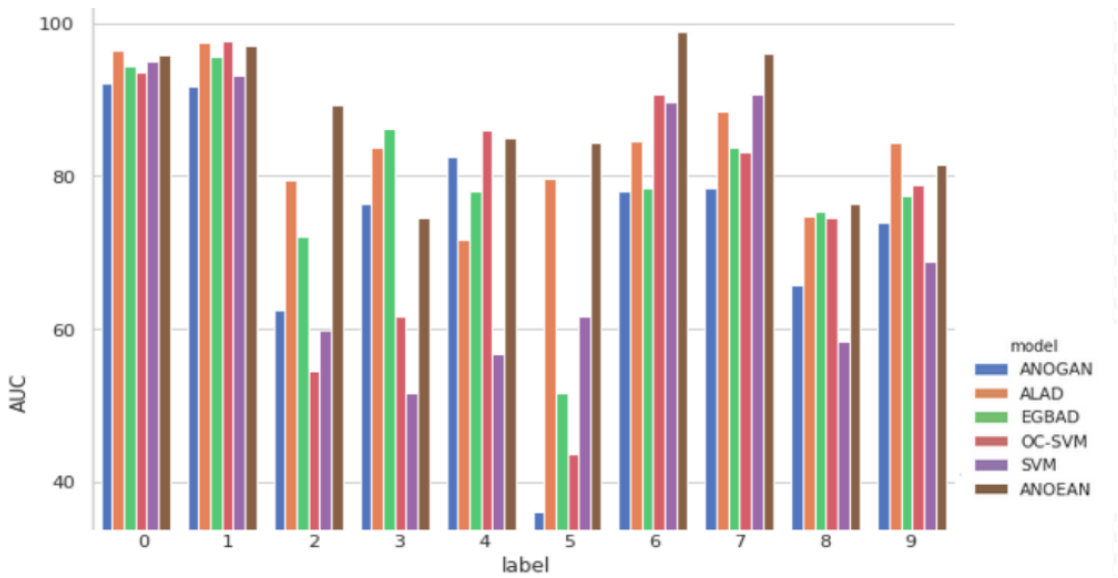


FIGURE 2 – Performance des modèles suivant la métrique AUC sur chaque chiffre dans le cas du problème MNIST [1 contre tous]

sique [16], nous avons rééquilibré les classes en donnant un poids plus élevé à la classe la moins fréquente (celle des anomalies).

AnoGAN. AnoGAN est utilisé avec les mêmes paramètres que les auteurs de [14]. Le critère de détection d'anomalies tient compte de la fonction de perte en reconstruction, et des sorties de la dernière couche cachée du discriminateur.

EGBAD. L'encodeur est utilisé pour projeter une donnée test x en un point z de l'espace latent, puis le générateur pour reconstruire une donnée artificielle x' à partir de z. Le critère d'anomalie compare x et x' d'une façon similaire à AnoGAN [20].

ALAD. Basé sur l'architecture ALICE [11], ALAD [21] améliore EGBAD en forçant explicitement l'encodeur et le générateur à être inverse l'un de l'autre au cours de l'apprentissage (par *cycle consistency*). C'est un des discrimateurs d'ALAD qui est utilisé pour détecter les anomalies.

La détection d'anomalies avec OCSVM, ALAD, AnoGAN et EGBAD sont des méthodes *one-class*. On utilise uniquement des données normales dans la phase d'apprentissage. Dans AnoEAN et SVM, on introduit en plus quelques exemples d'anomalies ce qui entraîne un déséquilibre des classes.

Les performances des algorithmes sont évaluées principalement par leur aire sous la courbe précision-rappel (AUC). Nous calculons également la mesure F1 optimale, ainsi que le taux de fausse alarme et le rappel associé. Pour chaque modèle, nous sélectionnons les paramètres d'apprentissage qui maximisent l'AUC calculée à partir du score d'anomalie $a(x)$ établi avec un ensemble de validation.

AnoEAN possède deux réseaux de neurones, l'encodeur et le discriminateur. Dans le cas de MNIST, l'encodeur est un réseau convolutif $(4*2*32, 4*2*64, 4*2*128, d)$ et le discriminateur est un réseau multi-couches $(32, 16, 1)$. Pour les bases NSL-KDD et KDD99, l'encodeur et le discriminateur sont des réseaux multi-couches $(128, 64, 32, d)$ et $(32, 16, 1)$ respectivement. La taille des *batch* est de 200. Le nombre d'*epoch* est fixé par *early stopping* et la descente du gradient sera avec Adam ($lr = 10^{-3}$).

3.2 Les résultats

Dans la Fig. 2, nous présentons nos résultats sur MNIST [1 contre tous] : chaque chiffre est successivement considéré comme la classe normale, les autres étant vus comme des anomalies. AnoEAN surpasse toutes les autres méthodes dans 5 cas sur 10 ; il a des performances comparables aux meilleures méthodes sur les 5 chiffres restants. Remarquons que dans l'absolu, toutes les méthodes ont des performances faibles sur les classes 2, 3, 4 et 5, car ces classes ont des caractéristiques visuelles similaires à celles des anoma-

lies. Nous observons aussi que les approches à base de reconstruction (ALAD et EGBAD) sont compétitives. Elles réussissent à recontruire correctement les chiffres, ce qui leur permet d'avoir un score d'anomalie important sur les éléments qui ne sont pas visuellement similaires à la classe normale.

Dans la Fig. 3, nous montrons les résultats sur MNIST [n contre m]. Nous nous intéressons au problème de l'identification des anomalies dans un ensemble de données où la classe normale est hétérogène (composée de plusieurs chiffres). Dans cette configuration, AnoEAN surpasse largement l'état de l'art, même si ses performances se dégradent avec l'hétérogénéité croissante de la classe normale. On remarque que les performances des approches à base de reconstruction (ALAD et EGBAD) s'effondrent. En effet, pour qu'elles fonctionnent, il est nécessaire que le générateur et l'encodeur soient inverses l'un de l'autre, ce qui est particulièrement difficile à assurer lorsque la classe normale est hétérogène. En revanche, AnoEAN réussit à encoder des distributions complexes car il ne necessite pas de calculer l'inverse de l'encodeur.

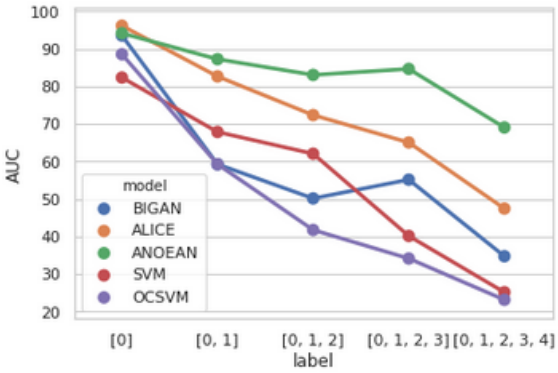


FIGURE 3 – Performances des différentes méthodes dans le cas du problème MNIST[n contre m]. La classe normale est constituée de 1 à 5 chiffres.

Les résultats sur les bases NSL-KDD et KDD99 sont présentés dans les Tables 1 et 2 . Outre l'AUC, nous montrons le taux de fausses alarmes (FA) qui est la proportion de fausses anomalies par rapport au nombre d'anomalies détectées, et le rappel qui est la proportion d'anomalies détectées par rapport à l'ensemble des anomalies de la base de test. Concernant le taux de fausses alarmes, la méthode SVM est celle qui a les meilleurs résultats sur KDD99 (car c'est un problème de classification), mais AnoEAN a des performances assez proches. Concernant le rappel, AnoEAN surpasse les autres méthodes, car nous avons explicitement codé dans sa fonction de coût le fait qu'il devait projeter toute anomalie en dehors de la zone des données normales dans l'espace latent.

AUC	F1	FA	Rappel	Modèle
79.6	80.5	19.6	79.6	AnoGAN
96.0	92.1	8.0	92.2	EGBAD
94.7	87.8	13.5	89.2	ALAD
94.1	87.9	13.1	89.0	OCSVM
97.5	**94.8**	**5.0**	94.6	**AnoEAN**
97.3	93.3	8.5	**95.2**	SVM

TABLE 1 – Résultats des différentes méthodes sur la base NSL-KDD

AUC	F1	FA	Rappel	Modèle
72.9	73.4	23.3	70.1	AnoGAN
89.1	93.9	5.5	93.3	EGBAD
95.4	96.6	3.6	96.6	ALAD
86.0	94.3	7.4	96.2	OCSVM
98.9	97.6	3.9	**99.0**	**AnoEAN**
98.6	98.7	**0.6**	97.6	SVM

TABLE 2 – Résultats des différentes méthodes sur la base KDD99

4 Conclusions et perspectives

Nous avons présenté une nouvelle méthode de détection d'anomalies qui utilise une projection dans un espace latent de faible dimension dans lequel exemples normaux et anomalies sont séparés. Pour cela, nous entrainons un encodeur par *adversarial learning*. La fonction de coût permet de modéliser les données normales dans dans un espace qui suit une distribution gaussienne, et les rares exemples d'anomalies en dehors de cette distribution. Notre méthode est moins couteuse, en terme de temps d'apprentissage, que les méthodes classiques basées sur les GAN, car elle ne nécessite pas d'apprendre un générateur. Pour la prédiction, notre méthode est également beaucoup plus rapide et légère en mémoire que les autres car elle necessite uniquement d'utiliser l'encodeur, ceci est un avantage pour toute application dans des systèmes embarqués. Nous avons montré expérimentalement que notre modèle peut faire face à des types d'anomalies non rencontrés au moment de l'apprentissage. De plus, notre technique a des meilleures performances que

l'état de l'art lorsqu'elle est confrontée à une classe normale hétérogène. Enfin, sur les bases habituelles utilisées en détection d'attaques dans des réseaux, notre technique déclenche peu de fausses alarmes. On note que les modèles entraînés d'une façon adversarial souffrent de problème de convergence, des nouvelles fonctions de coût et régularisation peuvent être rajouté pour stabilisé l'apprentissage. Il existe aussi un autre phénomène dit *Mode collapse*(problème d'effondrement), dans notre cas le *mode collapse* n'est pas contraignant, car cela signifie que l'encodeur projette toutes données normale dans un point point dans l'espace latent et et toutes données anormale dans un autre point éloigné dans le même espace, Dans se cas la distance entre les exemples normaux et les anomalies restera importante (score important). Parmi les pistes d'amélioration, nous travaillons à la suppression de tout exemple d'anomalie lors de l'apprentissage. Pour ce faire, une idée est d'apprendre un générateur d'anomalies, en utilisant l'encodeur et le discriminateur actuels comme adversaires.

Références

[1] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *Proc. ECML 2004*, pages 39–50, 2004.

[2] S. Akcay, A. A. Abarghouei, and T. P. Breckon. Ganomaly : Semi-supervised anomaly detection via adversarial training. *CoRR*, abs/1805.06725, 2018.

[3] R. Chalapathy and S. Chawla. Deep learning for anomaly detection : A survey. *CoRR*, abs/1901.03407, 2019.

[4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection : A survey. *ACM Comput. Surv.*, 41(3) :15 :1–15 :58, 2009.

[5] A. Creswell and A. A. Bharath. Inverting the generator of A generative adversarial network (II). *CoRR*, abs/1802.05701, 2018.

[6] L. Deecke, R. A. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft. Image anomaly detection with generative adversarial networks. In *Proc. ECML 2018*, pages 3–17, 2018.

[7] L. Dhanabal and S. Shantharajah. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. 2015.

[8] I. Golan and R. El-Yaniv. Deep anomaly detection using geometric transformations. In *Proc. NIPS 2018*, pages 9781–9791, 2018.

[9] B. R. Kiran, D. M. Thomas, and R. Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *J. Imaging*, 4(2) :36, 2018.

[10] Y. LeCun. The MNIST database of handwritten digits. 1998.

[11] C. Li, H. Liu, C. Chen, Y. Pu, L. Chen, R. Henao, and L. Carin. ALICE : towards understanding adversarial learning for joint distribution matching. In *Proc. NIPS 2017*, pages 5501–5509, 2017.

[12] M. Lichman. UCI machine learning repository,. 2013.

[13] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli. Adversarially learned one-class classifier for novelty detection. In *Proc. CVPR 2018*, pages 3379–3388, 2018.

[14] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *CoRR*, abs/1703.05921, 2017.

[15] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12(3) :582–588, 2000.

[16] K. Veropoulos, C. Campbell, and N. Cristianini. Controlling the sensitivity of support vector machines. *Proc. IJCAI 1999*, 1999.

[17] G. Wu and E. Y. Chang. Adaptive feature-space conformal transformation for imbalanced-data learning. In *Proc. ICML 2003*, pages 816–823, 2003.

[18] Y. Xia, X. Cao, F. Wen, G. Hua, and J. Sun. Learning discriminative reconstructions for unsupervised outlier removal. In *Proc. 2015 IEEE International Conference on Computer Vision (ICCV'15)*, pages 1511–1519, 2015.

[19] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao. Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications. *CoRR*, abs/1802.03903, 2018.

[20] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar. Efficient GAN-based anomaly detection. *CoRR*, abs/1802.06222, 2018.

[21] H. Zenati, M. Romain, C. Foo, B. Lecouat, and V. Chandrasekhar. Adversarially learned anomaly detection. In *Proc. ICDM 2018*, pages 727–736, 2018.

SpykeTorch: Efficient Simulation of Convolutional Spiking Neural Networks with at most one Spike per Neuron

Milad Mozafari[1], Mohammad Ganjtabesh[1], Abbas Nowzari-Dalini[1], and Timothée Masquelier*[2]

[1]Department of Computer Science, School of Mathematics, Statistics, and Computer Science, University of Tehran, Tehran, Iran
[2]CERCO UMR 5549, CNRS – Université Toulouse 3, France

May 27, 2019

Abstract

Application of deep convolutional spiking neural networks (SNNs) to artificial intelligence (AI) tasks has recently gained a lot of interest since SNNs are hardware-friendly and energy-efficient. Unlike the non-spiking counterparts, most of the existing SNN simulation frameworks are not practically efficient enough for large-scale AI tasks. In this methods paper, we introduce SpykeTorch, an open-source high-speed simulation framework based on PyTorch. This framework simulates convolutional SNNs with at most one spike per neuron and the rank-order encoding scheme. In terms of learning rules, both spike-timing-dependent plasticity (STDP) and reward-modulated STDP (R-STDP) are implemented, but other rules could be implemented easily. Apart from the aforementioned properties, SpykeTorch is highly generic and capable of reproducing the results of various studies. Computations in the proposed framework are tensor-based and totally done by PyTorch functions, which in turn brings the ability of just-in-time optimization for running on CPUs, GPUs, or Multi-GPU platforms.

Keywords: Convolutional spiking neural networks, Time-to-the-First-Spike coding, One spike per neuron, STDP, Reward-Modulated STDP, Tensor-Based computing, GPU acceleration.

1 Introduction

For many years, scientist were trying to bring human-like vision into machines and artificial intelligence (AI). In recent years, with advanced techniques based on deep convolutional neural networks (DCNNs), artificial vision has never been closer to human vision. Although DCNNs have shown outstanding results in many AI fields, they suffer from being data- and energy-hungry. Energy consumption is of vital importance when it comes to hardware implementation for solving real-world problems.

Our brain consumes much less energy than DCNNs, about 20 Watts [17] – roughly the power consumption of an average laptop, for its top-notch intelligence. This feature has convinced researchers to start working on computational models of human cortex for AI purposes. Spiking neural networks (SNNs) are the next generation of neural networks, in which neurons communicate through binary signals known as spikes. SNNs are energy-efficient for hardware implementation, because, spikes bring the opportunity of using event-based hardware as well as simple energy-efficient accumulators instead of complex energy-hungry multiply-accumulators that are usually employed in DCNN hardware [10, 5].

Spatio-temporal capacity of SNNs makes them potentially stronger than DCNNs, however, harnessing their ultimate power is not straightforward. Various types of SNNs have been proposed for vision tasks which can be categorized based on their specifications such as:

- network structure: shallow [16, 32, 14], and deep [19, 15],
- topology of connections: convolutional [3, 26], and fully connected [6],
- information coding: rate [3, 7], and latency [16, 6, 18],

*timothee.masquelier@cnrs.fr

- learning rule: unsupervised [6, 8, 27], supervised [7, 30, 18, 23, 33, 2], and reinforcement [9, 20].

For recent advances in deep learning with SNNs, we refer the readers to reviews by [25], [22], and [21].

Deep convolutional SNNs (DCSNNs) with time-to-first-spike information coding and STDP-based learning rule constitute one of those many types of SNNs that carry interesting features. Their deep convolutional structure supports visual cortex and let them extract features hierarchically from simple to complex. Information coding using the earliest spike time, which is proposed based on the rapid visual processing in the brain [28], needs only a single spike, making them super fast and more energy efficient. These features together with hardware-friendliness of STDP, turn this type of SNNs into the best option for hardware implementation and online on-chip training [31]. Several recent studies have shown the excellence of this type of SNNs in visual object recognition [20, 19, 15, 18].

With simulation frameworks such as Tensorflow and PyTorch, developing and running DCNNs is fast and efficient. Conversely, DCSNNs suffer from the lack of such frameworks. Existing state-of-the-art SNN simulators have been mostly developed for studying neuronal dynamics and brain functionalities and are not efficient and user-friendly enough for AI purposes. For instance, bio-realistic and detailed SNN simulations are provided by NEST [11], BRIAN [24], NEURON [4], and ANNarchy [29]. These frameworks also enable users to define their own dynamics of neurons and connections. In contrast, frameworks such as Nengo [1] and NeuCube [13] offer high-level simulations focusing on the neural behavior of the network. Recently, BindsNet [12] framework has been proposed as a fast and general SNN simulator based on PyTorch that is mainly developed for conducting AI experiments. A detailed comparison between BindsNet and the other available frameworks can be found in their paper.

In this paper, we propose SpykeTorch, a simulation framework based on PyTorch which is optimized specifically for convolutional SNNs with at most one spike per neuron. SpykeTorch offers utilities for building hierarchical feedforward SNNs with deep or shallow structures and learning rules such as STDP [16, 6, 15] and R-STDP [20, 19]. SpykeTorch only supports time-to-first-spike information coding and provides a non-leaky integrate and fire neuron model with at most one spike per stimulus. Unlike BindsNet which is flexible and general, the proposed framework is highly restricted to and optimized for this type of SNNs. Although BindsNet is based on PyTorch, its network design language is different. In contrast, SpykeTorch is fully compatible and integrated with PyTorch and obeys the same design language. Therefore, a PyTorch user may only read the documentation to find out the new functionalities. Besides, this integrity makes it possible to utilize almost all of the PyTorch's functionalities either running on a CPU, or (multi-) GPU platform.

The rest of this paper is organized as follows: Section 2 describes how SpykeTorch includes the concept of time in its computations. Section 3 is dedicated to SpykeTorch package structure and its components. In Section 4, a brief tutorial on building, training, and evaluating a DCSSN using SpykeTorch is given. Section 5 summarizes the current work and highlights possible future works.

2 Time Dimension

Modules in SpykeTorch are compatible with those in PyTorch and the underlying data-type is simply the PyTorch's tensors. However, since the simulation of SNNs needs the concept of "time", SpykeTorch considers an extra dimension in tensors for representing time. The user may not need to think about this new dimensionality while using SpykeTorch, but, in order to combine it with other PyTorch's functionalities or extracting different kinds of information from SNNs, it is important to have a good grasp of how SpykeTorch deals with time.

SpykeTorch works with time-steps instead of exact time. Since the neurons emit at most one spike per stimulus, it is enough to keep track of the first spike times (in time-step scale) of the neurons. For a particular stimulus, SpykeTorch divides all of the spikes into a pre-defined number of spike bins, where each bin corresponds to a single time-step. More precisely, assume a stimulus is represented by F feature maps, each constitutes a grid of $H \times W$ neurons. Let T_{max} be the maximum possible number of time-steps (or bins) and $T_{f,r,c}$ denote the spike time (or the bin index) of the neuron placed at position (r, c) of the feature map f, where $0 \leq f < F$, $0 \leq r < H$, $0 \leq c < W$, and $T_{f,r,c} \in \{0, 1, ..., T_{max} - 1\} \cup \{\infty\}$. The ∞ symbol stands for no spike. SpykeTorch considers this stimulus as a four-dimensional binary spike-wave tensor S of size $T_{max} \times F \times H \times W$ where:

$$S[t, f, r, c] = \begin{cases} 0 & t < T_{f,r,c}, \\ 1 & \textit{otherwise}. \end{cases} \tag{1}$$

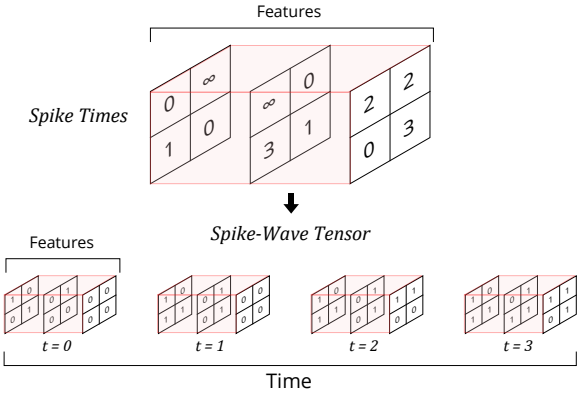


Figure 1: An example of generating spike-wave tensor from spike times. There are three feature maps, each constitutes a 2×2 grid of neurons that are going to generate a spike-wave representing a stimulus. If the maximum number of time-steps is 4, then the resulting spike-wave is a four-dimensional tensor of size $4 \times 3 \times 2 \times 2$. If a neuron emits spike at time step $t = i$, the corresponding position in the spike-wave tensor will be set to 1 from time-step $t = i$ to the final time step ($t = 3$).

Note that this way of keeping the spikes (accumulative structure) does not mean that neurons keep firing after their first spikes. Repeating spikes in future time steps increases the memory usage, but makes it possible to process all of the time-steps simultaneously and produce the corresponding outputs, which consequently results in a huge speed-up. Figure 1 illustrates an example of converting spike times into a SpykeTorch-compatible spike-wave tensor.

3 Package Structure

Basically, SpykeTorch consists of four python modules; (1) `snn` which contains multiple classes for creating SNNs, (2) `functional` that implements useful SNNs' functions, (3) `utils` which gathers helpful utilities, and (4) `visualization` which helps to generate graphical data out of SNNs. The following subsections explain these modules.

3.1 `snn` Module

The `snn` module contains necessary classes to build SNNs. These classes are inherited from the PyTorch's `nn.Module`, enabling them to function inside the PyTorch framework as network modules. Since we do not support error back-propagation, the PyTorch's auto-grad feature is turned off for all of the parameters in `snn` module.

`snn.Convolutional` objects implements spiking convolutional layers with two-dimensional convolution kernel. A `snn.Convolutional` object is built by providing the number of input and output features (or channels), and the size of the convolution kernel. Given the size of the kernel, the corresponding tensor of synaptic weights is randomly initialized using a normal distribution, where the mean and standard deviation can be set for each object, separately.

A `snn.Convolutional` object with kernel size $K_h \times K_w$ performs a valid convolution (with no padding) over an input spike-wave tensor of size $T_{max} \times F_{in} \times H_{in} \times W_{in}$ with stride equals to 1 and produces an output potentials tensor of size $T_{max} \times F_{out} \times H_{out} \times W_{out}$, where:

$$\begin{aligned} H_{out} &= H_{in} - K_h + 1, \\ W_{out} &= W_{in} - K_w + 1, \end{aligned} \tag{2}$$

and F_{in} and F_{out} are the number of input and output features, respectively. Potentials tensors (P) are similar to the binary spike-wave tensors, however $P[t, f, r, c]$ denotes the floating-point potential of a neuron placed at position (r, c) of feature map f, at time-step t.

The underlying computation of `snn.Convolutional` is the PyTorch's two-dimensional convolution, where the mini-batch dimension is sacrificed for the time. According to the accumulative structure of spike-wave tensor, the result of applying PyTorch's `conv2D` over this tensor is the accumulative potentials over time-steps.

Pooling is an important operation in deep convolutional networks. `snn.Pooling` implements two-dimensional max-pooling operation. Building `snn.Pooling` objects requires providing the pooling window size. The stride is equal to the window size by default, but it is adjustable. Zero padding is also another option which is off by default.

`snn.Pooling` objects are applicable to both spike-wave and potentials tensors. According to the structure of these tensors, if the input is a spike-wave tensor, then the output will contain the earliest spike within each pooling window, while if the input is a potentials tensor, the maximum potential within each pooling window will be extracted. Assume that the input tensor has the shape $T_{max} \times F_{in} \times H_{in} \times W_{in}$, the pooling window has the size $P_h \times P_w$ with stride $R_h \times R_w$, and the padding is (D_h, D_w), then the output tensor will have the size $T_{max} \times F_{out} \times H_{out} \times W_{out}$,

where:

$$H_{out} = \lfloor \frac{H_{in} + 2 \times D_h}{R_h} \rfloor, \quad W_{out} = \lfloor \frac{W_{in} + 2 \times D_w}{R_w} \rfloor. \tag{3}$$

To apply STDP on a convolutional layer, a `snn.STDP` object should be built by providing the value of required parameters such as learning rates. Since this simulator works with time-to-first-spike coding, the provided implementation of the STDP function is as follows:

$$\Delta \mathcal{W}_{i,j} = \begin{cases} A^+ \times (\mathcal{W}_{i,j} - LB) \times (UB - \mathcal{W}_{i,j}) & \textit{if} \quad T_j \leq T_i, \\ A^- \times (\mathcal{W}_{i,j} - LB) \times (UB - \mathcal{W}_{i,j}) & \textit{if} \quad T_j > T_i, \end{cases} \tag{4}$$

where, $\Delta \mathcal{W}_{i,j}$ is the amount of weight change of the synapse connecting the post-synaptic neuron i to the pre-synpatic neuron j, A^+ and A^- are learning rates, and $(\mathcal{W}_{i,j} - LB) \times (UB - \mathcal{W}_{i,j})$ is a stabilizer term which slows down the weight change when the synaptic weight $(\mathcal{W}_{i,j})$ is close to the lower (LB) or upper (UB) bounds.

To apply STDP during the training process, providing the input and output spike-wave, as well as output potentials tensors are necessary. `snn.STDP` objects make use of the potentials tensor to find winners. Winners are selected first based on the earliest spike times, and then based on the maximum potentials. The number of winners is set to 1 by default. `snn.STDP` objects also provide lateral inhibition, by which they completely inhibit the winners' surrounding neurons in all of the feature maps within a specific distance. This increases the chance of learning diverse features. Note that R-STDP can be applied using two `snn.STDP` objects; one for STDP part and the other for anti-STDP part.

3.2 `functional` Module

This module contains several useful and popular functions applicable on SNNs. Here we briefly review the most important ones. For the sake of simplicity, we replace the `functional.` with `sf.` for writing the function names.

As mentioned before, `snn.Convolutional` objects give potential tensors as their outputs. `sf.fire` takes a potentials tensor as input and converts it into a spike-wave tensor based on a given threshold. `sf.threshold` function is also available separately that takes a potentials tensor and outputs another potentials tensor in which all of the potentials lower than the given threshold are set to zero. The output of `sf.threshold` is called thresholded potentials.

Lateral inhibition is another vital operation for SNNs specially during the training process. It helps to learn more diverse features and achieve sparse representations in the network. SpykeTorch's `functional` module provides several functions for different kinds of lateral inhibitions.

`sf.feature_inhibition` is useful for complete inhibition of the selected feature maps. This function comes in handy to apply dropout to a layer. `sf.pointwise_inhibition` employs competition among features. In other words, at each location, only the neuron corresponding to the most salient feature will be allowed to emit a spike (the earliest spike with the highest potential). Lateral inhibition is also helpful to be applied on input intensities before conversion to spike-waves. This will decrease the redundancy in each region of the input. To apply this kind of inhibition, `sf.intensity_lateral_inhibition` is provided. It takes intensities and a lateral inhibition kernel by which it decreases the surrounding intensities (thus increases the latency of the corresponding spike) of each salient point. Local normalization is also provided by `sf.local_normalization` which uses regional mean for normalizing intensity values.

Winners-take-all (WTA) is a popular competition mechanism in SNNs. WTA is usually used for plasticity, however, it can be involved in other functionalities such as decision-making. `sf.get_k_winners` takes the desired number of winners and the thresholded potentials and returns the list of winners. Winners are selected first based on the earliest spike times, and then based on the maximum potentials. Each winner's location is represented by a triplet of the form $(feature, row, column)$.

3.3 `utils` Module

`utils` module provides several utilities to ease the implementation of ideas with SpykeTorch. For example, `utils.generate_inhibition_kernel` generates an inhibition kernel based on a series of inhibition factors in a form that can be properly used by `sf.intensity_lateral_inhibition`.

There exist several transformation utilities that are suitable for filtering inputs and converting them to spike-waves. Current utilities are mostly designed for vision

purposes. `utils.LateralIntencityInhibition` objects do the `sf.intensity_lateral_inhibition` as a transform object. `utils.FilterKernel` is a base class to define filter kernel generators. SpykeTorch has already provided `utils.DoGKernel` and `utils.GaborKernel` in order to generate DoG and Gabor filter kernels, respectively. Objects of `utils.FilterKernel` can be packed into a multi-channel filter kernel by `utils.Filter` objects and applied to the inputs.

The most important utility provided by `utils` is `utils.Intensity2Latency`. Objects of `utils.Intensity2Latency` are used as transforms in PyTorch's datasets to transform intensities into latencies, i.e. spike-wave tensors. Usually, `utils.Intensity2Latency` is the final transform applied to inputs.

Since the application of a series of transformations and the conversion to spike-waves can be time-consuming, SpykeTorch provides a wrapper class, called `utils.CacheDataset`, which is inherited from PyTorch's `dataset` class. Objects of `utils.CacheDataset` take a dataset as their input and cache the data after applying all of the transformations. They can cache either on primary memory or secondary storage.

Additionally, `utils` contains two functions `utils.tensor_to_text` and `utils.text_to_tensor`, which handle conversion of tensors to text files and the reverse, respectively. This conversion is helpful to import data from a source or export a tensor for a target software. The format of the text file is as follows: the first line contains comma-separated integers denoting the shape of the tensor. The second line contains comma-separated values indicating the whole tensor's data in row-major order.

3.4 `visualization` Module

The ability to visualize deep networks is of great importance since it gives a better understanding of how the network's components are working. However, visualization is not a straightforward procedure and depends highly on the target problem and the input data.

Due to the fact that SpykeTorch is developed mainly for vision tasks, its `visualization` module contains useful functions to reconstruct the learned visual features. The user should note that these functions are not perfect and cannot be used in every situation. In general, we recommend the user to define his/her own visualization functions to get the most out of the data.

4 Tutorial

In this section, we show how to design, build, train, and test a SNN with SpykeTorch in a tutorial format. The network in this tutorial is adopted from the deep convolutional SNN proposed by [19] which recognizes handwritten digits (tested on MNIST dataset). This network has a deep structure and uses both STDP and Reward-Modulated STDP (R-STDP), which makes it a suitable choice for a complete tutorial. In order to make the tutorial as simple as possible, we present code snippets with reduced contents. For the complete source code, please check SpykeTorch's GitHub[1] web page.

4.1 Step 1. Network Design

4.1.1 Structure

The best way to design a SNN is to define a class inherited from `torch.nn.Module`. The network proposed by [19], has an input layer which applies DoG filters to the input image and converts it to spike-wave. After that, there are three convolutional (S) and pooling (C) layers that are arranged in the form of $S1 \rightarrow C1 \rightarrow S2 \rightarrow C2 \rightarrow S3 \rightarrow C3$ (see Figure 2). Therefore, we need to consider three objects for convolutional layers in this model. For the pooling layers, we will use the functional version instead of the objects.

As shown in Listing 1, three `snn.Convolutional` objects are created with desired parameters. Two `snn.STDP` objects are built for training $S1$ and $S2$ layers. Since $S3$ is trained by R-STDP, two `snn.STDP` are needed to cover both STDP and anti-STDP parts. To have the effect of anti-STDP, it is enough to negate the signs of the learning rates. Note that the `snn.STDP` objects for `conv3` have two extra parameters where the first one turns off the stabilizer and the second one keeps the weights in range $[0.2, 0.8]$.

Although `snn` objects are compatible with `nn.Sequential` (`nn.Sequential` automates the forward pass given the network modules), we cannot use it at the moment. The reason is that different learning rules may need different kinds of data from each layer, thus accessing each layer during the forward pass is a must.

[1] `https://github.com/miladmozafari/SpykeTorch`

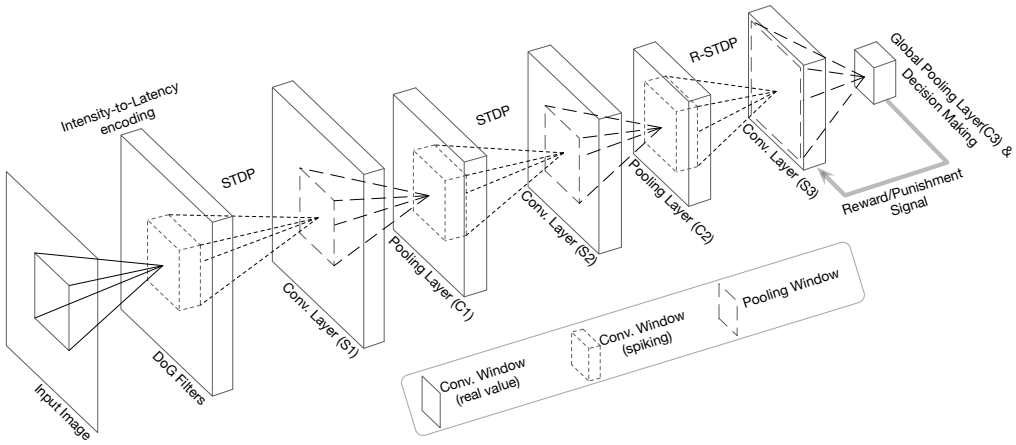


Figure 2: Overall structure of the network in the tutorial (modified figure from the work by [19]).

```python
import torch.nn as nn
import SpykeTorch.snn as snn
import SpykeTorch.functional as sf
class DCSNN(nn.Module):
  def __init__(self):
      super(DCSNN, self).__init__()

      #(in_channels, out_channels, kernel_size, weight_mean=0.8, weight_std=0.02)
      self.conv1 = snn.Convolution(6, 30, 5, 0.8, 0.05)
      self.conv2 = snn.Convolution(30, 250, 3, 0.8, 0.05)
      self.conv3 = snn.Convolution(250, 200, 5, 0.8, 0.05)

      #(conv_layer, learning_rate, use_stabilizer=True, lower_bound=0, upper_bound=1)
      self.stdp1 = snn.STDP(self.conv1, (0.004, -0.003))
      self.stdp2 = snn.STDP(self.conv2, (0.004, -0.003))
      self.stdp3 = snn.STDP(self.conv3, (0.004, -0.003), False, 0.2, 0.8)
      self.anti_stdp3 = snn.STDP(self.conv3, (-0.004, 0.0005), False, 0.2, 0.8)
```

Listing 1. Defining the network class.

```python
def forward(self, input):
    input = sf.pad(input, (2,2,2,2))
    if not self.training:
        pot = self.conv1(input)
        spk = sf.fire(pot, 15)
        pot = self.conv2(sf.pad(sf.pooling(spk, 2, 2), (1,1,1,1)))
        spk = sf.fire(pot, 10)
        pot = self.conv3(sf.pad(sf.pooling(spk, 3, 3), (2,2,2,2)))
        # omitting the threshold parameters means infinite threshold
        spk = sf.fire_(pot)
        winners = sf.get_k_winners(pot, 1)
        output = -1
        # each winner is a tuple of form (feature, row, column)
        if len(winners) != 0:
            output = self.decision_map[winners[0][0]]
        return output
```

Listing 2. Defining the forward pass (during testing process).

4.1.2 Forward Pass

Next, we implement the forward pass of the network. To this end, we override the `forward` function in `nn.Module`. If the training is off, then implementing the forward pass will be straightforward. Listing 2 shows the application of convolutional and pooling layers on an input sample. Note that each input is a spike-wave tensor. We will demonstrate how to convert images into spike-waves later.

As shown in Listing 2, the input of each convolutional layer is the padded version of the output of its previous layer, thus, there would be no information loss at the boundaries. Pooling operations are also applied by the corresponding function `sf.pooling`, which is an alternative to `snn.Pooling`. According to [19], their proposed network makes decision based on the maximum potential among the neurons in the last pooling layer. To this end, we use an infinite threshold for the last convolutional layer by omitting its value from `sf.fire_` function. `sf.fire_` is the in-place version of `sf.fire` which modifies the input potentials tensor P_{in} as follows:

$$P_{in}[t,f,r,c] = \begin{cases} 0 & \textit{if} \quad t < T_{max} - 1, \\ P_{in}[t,f,r,c] & \textit{otherwise.} \end{cases} \tag{5}$$

Consequently, the resulting spike-wave will be a tensor in which, all the values are zero except for those non-zero potential values in the last time-step.

```
def save_data(self, input_spk, pot, spk, winners):
    self.ctx["input_spikes"] = input_spk
    self.ctx["potentials"] = pot
    self.ctx["output_spikes"] = spk
    self.ctx["winners"] = winners
```

Listing 3. Saving required data for plasticity.

Now that we have the potentials of all the neurons in $S3$, we find the only one winner among them. This is the same as doing a global max-pooling and choosing the maximum potential among them. `decision_map` is a Python list which maps each feature to a class label. Since each winner contains the feature number as its first component, we can easily indicate the decision of the network by putting that into the `decision_map`.

We cannot take advantage of this forward pass during the training process as the STDP and R-STDP need local synaptic data to operate. Therefore, we need to save the required data during the forward pass. We define a Python dictionary (named `ctx`) in our network class and a function which saves the data into that (see Listing 3). Since the training process is layer-wise, we update the `forward` function to take another parameter which specifies the layer that is under training. The updated `forward` function is shown in Listing 4.

There are several differences with respect to the testing forward pass. First, `sf.fire` is used with an extra parameter value. If the value of this parameter is `True`, the tensor of thresholded potentials will also be returned. Second, `sf.get_k_winners` is called with a new parameter value which controls the radius of lateral inhibition. Third, the forward pass is interrupted by the value of `max_layer`.

4.1.3 Plasticity

Now that we saved the required data for learning, we can define a series of helper functions to apply STDP or anti-STDP. Listing 5 defines three member functions for this purpose. For each call of `STDP` objects, we need to provide tensors of input spike-wave, output thresholded potentials, output spike-wave, and the list of winners.

4.2 Step 2. Input Layer and Transformations

SNNs work with spikes, thus, we need to transform images into spike-waves before feeding them into the network. PyTorch's datasets accept a function as a transformation

```
def forward(self, input, max_layer):
    input = sf.pad(input, (2,2,2,2))
    if self.training: #forward pass for train
        pot = self.conv1(input)
        spk, pot = sf.fire(pot, 15, True)
        if max_layer == 1:
            winners = sf.get_k_winners(pot, 5, 3)
            self.save_data(input, pot, spk, winners)
            return spk, pot
        spk_in = sf.pad(sf.pooling(spk, 2, 2),
        ↪  (1,1,1,1))
        pot = self.conv2(spk_in)
        spk, pot = sf.fire(pot, 10, True)
        if max_layer == 2:
            winners = sf.get_k_winners(pot, 8, 2)
            self.save_data(spk_in, pot, spk, winners)
            return spk, pot
        spk_in = sf.pad(sf.pooling(spk, 3, 3),
        ↪  (2,2,2,2))
        pot = self.conv3(spk_in)
        spk = sf.fire_(pot)
        winners = sf.get_k_winners(pot, 1)
        self.save_data(spk_in, pot, spk, winners)
        output = -1
        if len(winners) != 0:
            output = self.decision_map[winners[0][0]]
        return output
    else:
        # forward pass for testing process
```

Listing 4. Defining the forward pass (during training process).

which is called automatically on each input sample. We make use of this feature together with the provided transform functions and objects by PyTorch and SpykeTorch. According to the network proposed by [19], each image is convolved by six DoG filters, locally normalized, and transformed into spike-wave. As appeared in Listing 6, a new class is defined to handle the required transformations.

Each `InputTransform` object converts the input image into a tensor (line 9), adds an extra dimension for time (line 10), applies provided filters (line 11), applies local normalization (line 12), and generates spike-wave tensor (line 13). To create `utils.Filter` object, six DoG kernels with desired parameters are given to `utils.Filter`'s constructor (lines 15-17) as well as an appropriate padding and threshold value (line 18).

4.3 Step 3. Data Preparation

Due to the PyTorch and SpykeTorch compatibility, all of the PyTorch's dataset utilities work here. As illustrated in Listing 7, we use `torchvision.datasets.MNIST` to load MNIST dataset with our previously defined `transform`. Moreover, we use SpykeTorch's dataset wrapper, `utils.CacheDataset` to enable caching the trans-

```
def stdp(self, layer_idx):
    if layer_idx == 1:
        self.stdp1(self.ctx["input_spikes"],
        ↪  self.ctx["potentials"],
        ↪  self.ctx["output_spikes"],
        ↪  self.ctx["winners"])
    if layer_idx == 2:
        self.stdp2(self.ctx["input_spikes"],
        ↪  self.ctx["potentials"],
        ↪  self.ctx["output_spikes"],
        ↪  self.ctx["winners"])

def reward(self):
    self.stdp3(self.ctx["input_spikes"],
    ↪  self.ctx["potentials"],
    ↪  self.ctx["output_spikes"], self.ctx["winners"])

def punish(self):
    self.anti_stdp3(self.ctx["input_spikes"],
    ↪  self.ctx["potentials"],
    ↪  self.ctx["output_spikes"], self.ctx["winners"])
```

Listing 5. Defining helper functions for plasticity.

formed data after its first presentation. When the dataset gets ready, we use PyTorch's `DataLoader` to manage data loading.

4.4 Step 4. Training and Testing

4.4.1 Unsupervised Learning (STDP)

To do unsupervised learning on $S1$ and $S2$ layers, we use a helper function as defined in Listing 8. This function trains layer `layer_idx` of `network` on `data` by calling the corresponding `STDP` object. There are two important things in this function: (1) putting the network in train mode by calling `.train` function, and (2) loading the sample on GPU if the global `use_cuda` flag is `True`.

4.4.2 Reinforcement Learning (R-STDP)

To apply R-STDP, it is enough to call previously defined `reward` or `punish` member functions under appropriate conditions. As shown in Listing 9, we check the network's decision with the label and call `reward` (or `punish`) if it matches (or mismatches) the target. We also compute the performance by counting correct, wrong, and silent (no decision is made because of receiving no spikes) samples.

4.4.3 Execution

Now that we have the helper functions, we can make an instance of the network and start training and testing it. Listing 10 illustrates the implementation of this part. Note that the `test` helper function is the same as the

```
import SpykeTorch.utils as utils
import torchvision.transforms as transforms
class InputTransform:
    def __init__(self, filter):
        self.to_tensor = transforms.ToTensor()
        self.filter = filter
        self.temporal_transform =
        ↪  utils.Intensity2Latency(15, to_spike=True)
    def __call__(self, image):
        image = self.to_tensor(image) * 255
        image.unsqueeze_(0)
        image = self.filter(image)
        image = sf.local_normalization(image, 8)
        return self.temporal_transform(image)

kernels = [ utils.DoGKernel(3,3/9,6/9),
↪  utils.DoGKernel(3,6/9,3/9),
        utils.DoGKernel(7,7/9,14/9),
        ↪  utils.DoGKernel(7,14/9,7/9),
        utils.DoGKernel(13,13/9,26/9),
        ↪  utils.DoGKernel(13,26/9,13/9)]
filter = utils.Filter(kernels, padding = 6, thresholds =
↪  50)
transform = InputTransform(filter)
```

Listing 6. Transforming each input image into spike-wave.

```
from torch.utils.data import DataLoader
from torchvision.datasets import MNIST
MNIST_train = utils.CacheDataset(MNIST(root=data_root,
↪  train=True, download=True, transform=transform))
MNIST_test = utils.CacheDataset(MNIST(root=data_root,
↪  train=False, download=True, transform=transform))
MNIST_loader = DataLoader(MNIST_train, batch_size=1000)
MNIST_test_loader = DataLoader(MNIST_test,
↪  batch_size=len(MNIST_test))
```

Listing 7. Preparing MNIST dataset and the data loader.

`train_rl` function, but it calls `network.eval` instead of `network.train` and it does not call plasticity member functions. Also, invoking `net.cuda`, transfers all the network's parameters to the GPU.

4.5 Source Code

Through this tutorial, we omitted many parts of the actual implementation such as adaptive learning rates, multiplication of learning rates, and saving/loading the best state of the network, for the sake of simplicity and clearance. The complete reimplementation is available on SpykeTorch's GitHub web page. It is worth mentioning that this reimplementation achieved 97% of recognition performance which is almost the same as the original implementation. We have also developed a reimplementation of the networks in ref. [20, 15] and achieved almost the same results. However, due to technical and computational differences between SpykeTorch and the original versions, tiny differences in

```
def train_unsupervised(network, data, layer_idx):
  network.train()
  for i in range(len(data)):
     data_in = data[i].cuda() if use_cuda else data[i]
     network(data_in, layer_idx)
     network.stdp(layer_idx)
```

Listing 8. Helper function for unsupervised learning.

```
import numpy as np
def train_rl(network, data, target):
  network.train()
  perf = np.array([0,0,0]) # correct, wrong, silent
  for i in range(len(data)):
     data_in = data[i].cuda() if use_cuda else data[i]
     target_in = target[i].cuda() if use_cuda else
     ↪  target[i]
     d = network(data_in, 3)
     if d != -1:
        if d == target_in:
           perf[0]+=1
           network.reward()
        else:
           perf[1]+=1
           network.punish()
     else:
        perf[2]+=1
  return perf/len(data)
```

Listing 9. Helper function for reinforcement learning.

performance are expected.

5 Conclusions

In recent years, SNNs have gained many interests in AI because of their ability to work in a spatio-temporal domain as well as energy efficiency. Unlike DCNNs, most of the current SNN simulators are not efficient enough to perform large-scale AI tasks. In this paper, we proposed SpykeTorch, an open-source high-speed simulation framework based on PyTorch. The proposed framework is optimized for convolutional SNNs with at most one spike per neuron and time-to-first-spike information coding scheme. SpykeTorch provides STDP and R-STDP learning rules but other rules can be added easily.

The compatibility and integrity of SpykeTorch with PyTorch have simplified its usage specially for the deep learning communities. This integration brings almost all of the PyTorch's features functionalities to SpykeTorch such as the ability of just-in-time optimization for running on CPUs, GPUs, or Multi-GPU platforms.

We provided a tutorial on how to build, train, and evaluate a DCSNN for digit recognition using SpykeTorch. However, the resources are not limited to this paper and

```
net = DCSNN()
if use_cuda:
  net.cuda()
# First Layer
for epoch in range(epochs_1):
  for data,targets in MNIST_loader:
     train_unsupervised(net, data, 1)
# Second Layer
for epoch in range(epochs_2):
  for data,targets in MNIST_loader:
     train_unsupervised(net, data, 2)
# Third Layer
for epoch in range(epochs_3):
  for data,targets in MNIST_loader: # Training
     print(train_rl(net, data, targets))
  for data,targets in MNIST_test_loader: # Testing
     print(test(net, data, targets))
```

Listing 10. Training and testing the network.

additional scripts and documentations can be found on SpykeTorch's GitHub page. We reimplemented our previous works [20, 19] by SpykeTorch and reproduced their results with negligible difference.

Although the current version of SpykeTorch is functional and provides the main modules and utilities for DCSNNs (with at most one spike per neuron), we will not stop here and our plan is to extend and improve it gradually. For example, adding automation utilities would ease programming the network's forward pass resulting a more readable and cleaner code. Due to the variations of training strategies, designing a general automation platform is challenging. Another feature that improves SpykeTorch's speed is batch processing. Enabling batch mode might be easy for operations like convolution or pooling, however, implementing batch learning algorithms that can be run with none or a few CPU-GPU interactions is hard. Finally, implementing features to support models for other modalities such as the auditory system makes SpykeTorch a multi-modal SNN framework.

References

[1] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith. Nengo: a python tool for building large-scale functional brain models. *Frontiers in neuroinformatics*, 7:48, 2014.

[2] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, pages 795–805, 2018.

[3] Y. Cao, Y. Chen, and D. Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015.

[4] N. T. Carnevale and M. L. Hines. *The NEURON book*. Cambridge University Press, 2006.

[5] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[6] P. U. Diehl and M. Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9:99, 2015.

[7] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

[8] P. Ferré, F. Mamalet, and S. J. Thorpe. Unsupervised feature learning with winner-takes-all based stdp. *Frontiers in computational neuroscience*, 12:24, 2018.

[9] R. V. Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 2007.

[10] S. Furber. Large-scale neuromorphic computing systems. *Journal of neural engineering*, 13(5):051001, 2016.

[11] M.-O. Gewaltig and M. Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007.

[12] H. Hazan, D. J. Saunders, H. Khan, D. T. Sanghavi, H. T. Siegelmann, and R. Kozma. Bindsnet: A machine learning-oriented spiking neural networks library in python. *Frontiers in neuroinformatics*, 12:89, 2018.

[13] N. K. Kasabov. Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*, 52:62–76, 2014.

[14] S. R. Kheradpisheh, M. Ganjtabesh, and T. Masquelier. Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition. *Neurocomputing*, 205:382–392, 2016.

[15] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018.

[16] T. Masquelier and S. J. Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Computational Biology*, 3(2):e31, 2007.

[17] J. W. Mink, R. J. Blumenschine, and D. B. Adams. Ratio of central nervous system to body metabolism in vertebrates: its constancy and functional basis. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 241(3):R203–R212, 1981.

[18] H. Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7):3227–3235, 2018.

[19] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier. Bio-inspired digit recognition using spike-timing-dependent plasticity (stdp) and reward-modulated stdp in deep convolutional networks. *arXiv preprint arXiv:1804.00227*, 2019.

[20] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh. First-spike-based visual categorization using reward-modulated stdp. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):6178–6190, Dec 2018.

[21] E. O. Neftci, H. Mostafa, and F. Zenke. Surrogate gradient learning in spiking neural networks. *arXiv preprint arXiv:1901.09948*, 2019.

[22] M. Pfeiffer and T. Pfeil. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in neuroscience*, 12, 2018.

[23] S. B. Shrestha and G. Orchard. Slayer: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*, pages 1419–1428, 2018.

[24] M. Stimberg, D. F. Goodman, V. Benichoux, and R. Brette. Equation-oriented specification of neural models for simulations. *Frontiers in neuroinformatics*, 8:6, 2014.

[25] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida. Deep learning in spiking neural networks. *Neural Networks*, 2018.

[26] A. Tavanaei and A. S. Maida. Bio-inspired spiking convolutional neural network using layer-wise sparse coding and STDP learning. *arXiv preprint arXiv:1611.03000*, 2016.

[27] J. C. Thiele, O. Bichler, and A. Dupret. Event-based, timescale invariant unsupervised online deep learning with stdp. *Frontiers in computational neuroscience*, 12:46, 2018.

[28] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *nature*, 381(6582):520, 1996.

[29] J. Vitay, H. Ü. Dinkelbach, and F. H. Hamker. Annarchy: a code generation approach to neural simulations on parallel hardware. *Frontiers in neuroinformatics*, 9:19, 2015.

[30] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12, 2018.

[31] A. Yousefzadeh, T. Masquelier, T. Serrano-Gotarredona, and B. Linares-Barranco. Hardware implementation of convolutional stdp for on-line visual feature learning. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*, pages 1–4. IEEE, 2017.

[32] Q. Yu, H. Tang, K. C. Tan, and H. Li. Rapid feedforward computation by temporal encoding and learning with spiking neurons. *IEEE transactions on neural networks and learning systems*, 24(10):1539–1552, 2013.

[33] F. Zenke and S. Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

Apprentissage en ligne par renforcement centré utilisateur pour la composition émergente d'applications ambiantes

Walid Younes*[1], Françoise Adreit[2], Sylvie Trouilhet[1] et Jean-Paul Arcangeli[1]

[1]Université de Toulouse, IRIT, Université Paul Sabatier
[2]Université de Toulouse, IRIT, Université Toulouse Jean Jaurès

Résumé

Les systèmes cyber-physiques connectés et ambiants entourent l'utilisateur humain de services mis à sa disposition. Ces services, plus ou moins complexes, doivent être le plus possible adaptés à ses préférences et à la situation courante. Nous proposons de les construire automatiquement et à la volée par composition de services plus élémentaires présents sur le moment dans l'environnement, ceci sans expression préalable des besoins de l'utilisateur. Pour cela, avec la forte variabilité dynamique de l'environnement ambiant et des besoins, ce dernier doit être sollicité mais *a minima*. Afin de produire les connaissances nécessaires à la composition automatique et en l'absence de données initiales, nous définissons une solution d'apprentissage en ligne par renforcement à horizon infini. Cette solution apprend incrémentalement de l'utilisateur et pour l'utilisateur. Elle est décentralisée au sein d'un système multi-agent chargé de l'administration et de la composition des services.

Mots-clés : apprentissage en ligne, apprentissage par renforcement, feedback utilisateur, services logiciels, composition de services, système multi-agent, intelligence ambiante

1 Introduction

Les systèmes ambiants et mobiles sont composés d'appareils fixes ou mobiles reliés par un ou plusieurs réseaux de communication. Ces appareils hébergent des composants logiciels qui fournissent des services et qui peuvent, eux-mêmes, requérir d'autres services. Ces composants sont des briques logicielles qui peuvent être assemblées en connectant des services requis à des services fournis pour composer des applications plus complexes [Som16]. Par exemple, l'assemblage d'un composant d'interaction non dédié présent dans un smartphone (*e.g.*, un *curseur*, un *bouton* ou un composant de reconnaissance vocale), d'un *adaptateur* et d'une *lampe connectée* peut permettre de réaliser une application permettant à un utilisateur de contrôler l'éclairage ambiant.

Les composants (matériels et logiciels) sont en général multi-propriétaires et gérés de manière indépendante : ils sont développés, installés et activés indépendamment les uns des autres. En raison de la mobilité des appareils et des utilisateurs, ils peuvent apparaître ou disparaître selon une dynamique imprévisible, conférant aux systèmes ambiants et mobiles un caractère ouvert et instable. A cela s'ajoute le nombre souvent important de composants, source de difficultés lors du passage à l'échelle. Dans ce contexte, les assemblages de composants sont difficiles à concevoir, à entretenir et à adapter.

Plongé au sein de ces systèmes, l'utilisateur humain peut utiliser les services qui sont à sa disposition. L'intelligence ambiante [Wei91, CAJ09, Sad11] vise à lui offrir un environnement personnalisé, adapté à la situation, c'est-à-dire à lui fournir les bons services au bon moment, en anticipant ses besoins qui eux-mêmes peuvent évoluer. Pour cela, l'utilisateur peut être sollicité mais dans une limite raisonnable [BS03].

Notre projet a pour objectif de concevoir et de réaliser un système qui assemble dynamiquement et automatiquement les composants logiciels afin de construire des applications "composites" adaptées à l'environnement ambiant et à l'utilisateur, c'est-à-dire opérationnelles, utiles et utilisables [Cou13]. Notre approche rompt avec le traditionnel mode *top-down* pour le développement d'applications : la réalisation d'un assemblage n'est pas guidée par les besoins explicites de l'utilisateur ni par des plans d'assemblage prédéfinis ; au contraire, les application composites sont

*Walid.Younes@irit.fr

construites à la volée en mode *bottom-up* à partir des services présents sur le moment dans l'environnement ambiant. Ainsi, les applications émergent de l'environnement, en tirant profit des opportunités. Dans ce cadre, l'utilisateur ne demande pas un service ou une application : les applications émergentes lui sont fournies en mode *push* [1].

Notre solution repose sur un *middleware*, appelé moteur de composition, qui détecte périodiquement les composants présents dans l'environnement ambiant, conçoit des assemblages de composants de manière opportuniste et les propose à l'utilisateur. En l'absence de besoin explicité *a priori*, le moteur apprend les préférences de l'utilisateur en fonction de la situation.

Dans ce travail, nous avons développé une solution basée sur l'apprentissage automatique pour construire et adapter dynamiquement un environnement ambiant en fonction de l'utilisateur. L'objectif de cet article est de présenter les principes de cet apprentissage et de leur mise en œuvre.

L'article est organisé comme suit. L'architecture du système incluant le moteur de composition et l'utilisateur est présentée dans la section 2. Dans la section 3, notre problématique d'apprentissage est analysée en termes de motivations, d'objectifs et de données. Cette section se conclut par la définition du type d'apprentissage, à savoir en ligne et par renforcement. Les principes de notre solution d'apprentissage sont ensuite exposés dans la section 4. La section 5 résume l'état de l'art en matière d'apprentissage pour la composition logicielle automatique et positionne notre proposition de solution. En conclusion, dans la section 6, le lecteur trouvera un bref résumé de la contribution, un point d'avancement sur le développement de la solution et une discussion sur les problèmes ouverts et la suite de ce travail.

2 Architecture du système de composition

Afin de répondre à l'exigence d'automatisation de la composition opportuniste de composants logiciels, nous avons défini une architecture logicielle du système de composition [YTAA18]. Cette section en présente les grandes lignes et la figure 1 en donne une vue simplifiée.

Au cœur du système, le moteur OCE (*Opportunistic Composition Engine*) a pour fonction de concevoir pour l'utilisateur des applications composites en assemblant

1. Un exemple de cas d'utilisation est développé dans [YTAA18].

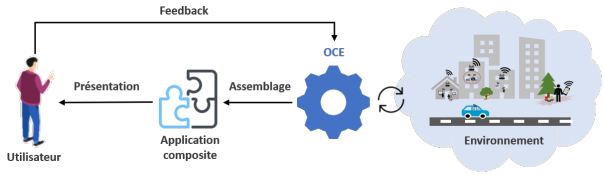


FIGURE 1 – Vue simplifiée de l'architecture

des composants métiers et des composants d'interaction disponibles. Pour cela, il perçoit les composants et leurs services présents dans l'environnement ambiant, puis choisit d'établir des connexions entre des services requis et des services fournis faisant ainsi émerger les applications.

OCE propose à l'utilisateur les applications émergentes et c'est l'utilisateur qui décide *in fine* de leur déploiement : il peut accepter ou rejeter une application proposée ou encore la modifier [KTAB18]. Ce point est particulièrement important dans le domaine de l'interaction homme-machine pour lequel le contrôle par l'utilisateur de son environnement d'interaction est fondamental [BS03]. Après acceptation, l'application émergente est déployée automatiquement.

Afin de solliciter l'utilisateur *a minima*, le système de composition doit être le plus autonome possible. Notre solution se conforme aux principes de l'informatique autonome et au modèle MAPE-K (*Monitor, Analyse, Plan, Execute - Knowledge*) [KC03] : de manière cyclique, OCE observe l'environnement ambiant, l'analyse et planifie des assemblages en se basant sur des connaissances. Dans le modèle MAPE-K, la phase d'exécution a pour fonction de réaliser ce qui a été préalablement planifié. Ici, l'application émergente est présentée à l'utilisateur sous la forme d'un assemblage de composants avant d'être (éventuellement) déployée. Les actions que l'utilisateur effectue en réponse à cette présentation sont sources de feedback et permettent à OCE de faire évoluer ses connaissances par apprentissage (voir les sections suivantes).

Comme, fondamentalement, l'environnement ambiant est distribué et les composants matériels et logiciels sont administrés par différentes autorités, nous avons choisi de concevoir le moteur sous la forme d'un système multi-agent (SMA) dans lequel chaque service d'un composant est "administré" par un agent. De manière générale, les agents sont des entités autonomes qui coopèrent afin d'atteindre un but commun [Fer99]. Ceux du moteur OCE ont pour rôle d'établir des connexions (ou des déconnexions) entre

les services qu'ils administrent. Pour cela, ils échangent des messages dans le cadre du protocole d'interaction ARSA (*Advertise, Reply, Select, Agree*) [YTAA18] : un agent annonce le service (fourni ou requis) qu'il administre dans le but de trouver un service partenaire avec qui se connecter, répond à des annonces s'il est intéressé par une connexion, sélectionne une ou des offres de connexion parmi les réponses reçues et accepte la sélection par un autre agent. Ce protocole supporte la coopération entre agents dans un contexte asynchrone, dynamique et ouvert. Les agents continuent de coopérer même si un agent disparaît (en cas de désactivation du service administré ou de panne) ou si un message n'arrive pas à son destinataire.

3 Apprendre : pourquoi, quoi, à partir de quoi et comment ?

La fonction du moteur OCE est de concevoir des assemblages de composants logiciels qui réalisent des applications émergentes pertinentes, puis de les proposer à l'utilisateur et enfin de les déployer suivant les retours de l'utilisateur. Pour cela, le moteur doit prendre des décisions de connexion entre services dans l'objectif de satisfaire l'utilisateur.

Plusieurs éléments contraignent ces prises de décision :

— la dynamique et l'imprévisibilité de l'environnement ambiant,
— le nombre de composants qui peut s'avérer important,
— la dynamique et la variabilité des besoins de l'utilisateur,
— la nécessité de solliciter *a minima* l'utilisateur.

Pour prendre des décisions pertinentes, le moteur a besoin de connaissances qui, dans ce contexte, doivent être apprises automatiquement.

Dans cette section, nous analysons pourquoi le moteur OCE doit apprendre, ce qu'il doit apprendre et quelles sont les données d'apprentissage.

3.1 Pourquoi apprendre ?

De par la dynamique et l'imprévisibilité de l'environnement ambiant, la combinatoire générée par le nombre important de composants ainsi que la dynamique et la variabilité de ses propres besoins, l'utilisateur n'est pas en mesure d'exprimer *a priori*, explicitement et d'une manière exhaustive ses besoins et ses préférences ni de les traduire en plans d'assemblages dans les différents situations qu'il peut rencontrer.

Par conséquent, OCE ne peut se baser ni sur des besoins de l'utilisateur explicités *a priori* ni sur des règles d'assemblage prédéfinies. OCE doit donc apprendre et ce, à partir de l'expérience.

3.2 Apprendre quoi ?

Le moteur OCE doit construire les connaissances nécessaires pour pouvoir proposer des applications pertinentes à l'utilisateur. Il doit apprendre ce que celui-ci préfère quand certains composants sont présents dans l'environnement ambiant. Basées sur ces préférences à un moment donné, ces connaissances seront exploitées pour prendre de futures décisions dans des situations identiques ou similaires. Par exemple, OCE peut apprendre que l'utilisateur préfère contrôler l'éclairage ambiant au moyen du curseur embarqué sur son smartphone plutôt qu'avec l'interrupteur mural connecté.

3.3 À partir de quoi apprendre ?

Si le moteur ne peut pas disposer initialement d'un ensemble de données d'apprentissage, en revanche, des données peuvent être observées au fil de l'utilisation du système pour servir de base aux décisions futures.

Nous distinguons plusieurs sources possibles :

1. L'utilisateur étant présent dans la boucle de contrôle, il est possible d'observer et d'exploiter ses retours sur l'application émergente proposée (acceptation, modification ou rejet) sans le surcharger. Par exemple, en cas de modification par l'utilisateur du contrôle de l'éclairage ambiant, OCE peut faire évoluer ses préférences de connexion entre services (le curseur du smartphone plutôt que l'interrupteur).
2. Pour modifier l'assemblage de composants proposé, l'utilisateur dispose d'un éditeur qui lui permet de manipuler les composants et leurs connexions [KTAB18]. En instrumentant l'éditeur, il serait possible d'observer certaines actions de l'utilisateur (par exemple, pousser un composant en dehors de la fenêtre d'édition) et d'en extraire un feedback complémentaire.
3. Au prix d'un effort supplémentaire, l'utilisateur pourrait explicitement donner un feedback plus riche sur l'application proposée, avant son déploiement ou après son utilisation.
4. Après déploiement, l'application peut aussi être observée (utilisation des composants participants, des services ou des connexions...) afin d'extraire automatiquement une appréciation qualitative de son utilisation.

5. En interne au SMA, il est également envisageable d'extraire des informations de feedback à partir des interactions entre les agents (par exemple, en cas d'échanges soutenus entre deux agents, on pourrait favoriser la connexion entre leurs services respectifs).

À ce stade de notre travail, nous avons choisi d'exploiter la première source de données qui traduit les préférences de l'utilisateur en fonction des composants présents dans l'environnement ambiant. Ainsi, le moteur apprend des interactions avec l'utilisateur sans le surcharger. Nous faisons l'hypothèse que même si l'utilisateur ne peut pas expliciter *a priori* ses besoins, il est capable de réagir sur le moment à la proposition d'une application construite automatiquement. Il est alors possible de capturer cette réaction sous la forme de feedback et d'en extraire de la connaissance utile à des prises de décision futures.

3.4 Comment apprendre ?

L'absence de données initiales et de solutions connues rend impossible un apprentissage supervisé ou non supervisé. De plus, la dynamique de l'environnement, avec les services qui apparaissent et disparaissent de manière imprévisible, rend très difficile voire impossible la construction d'un modèle statique de prédiction ou de classification. Pour ces raisons, nous proposons un apprentissage par adaptation progressive permettant au moteur OCE d'apprendre en continu en exploitant les informations de feedback que l'utilisateur fournit itérativement. C'est un apprentissage *en ligne, par renforcement et à horizon infini*. Ce type de solution permet à un système de s'adapter sur le long terme en interagissant avec son environnement [Boe14].

L'objet de l'apprentissage est ici de contribuer à déterminer une action. Les agents d'OCE raisonnent en s'appuyant sur des connaissances construites et mises à jour en ligne, de manière incrémentale, au fur et à mesure de l'expérience et au gré des interactions avec l'utilisateur et des éventuelles évolutions de ses préférences. Selon le modèle de l'apprentissage en ligne [CMB18], les agents font une "prédiction" (l'assemblage) et l'environnement (ici l'utilisateur) apporte une réponse. Cependant, le retour donné ici par l'utilisateur n'a pas le caractère d'exactitude de la réponse de l'environnement dans le modèle de l'apprentissage en ligne. Pour cette raison, nous hybridons les principes de l'apprentissage en ligne avec ceux de l'apprentissage par renforcement [CMB18] : la réponse de l'utilisateur permet de renforcer les connaissances des agents, et les décisions à l'itération t s'appuient sur les connaissances cumulées lors des itérations précédentes.

Enfin, considérant la dynamique, à la fois de l'environnement ambiant et de l'utilisateur, cet apprentissage doit être en plus à horizon infini [CMB18], ce qui n'exclut pas des phases de stabilisation des connaissances.

Notons en complément que cette approche n'exclut pas non plus la possibilité d'exploiter des connaissances connues *a priori* (par exemple, des règles générales d'assemblage de composants métiers, des règles d'ergonomie d'assemblage de composants d'interaction) qui pourraient être fournies initialement et ainsi accélérer le processus d'acquisition des connaissances.

4 Principes de la solution

Le moteur OCE est un système multi-agent dans lequel chaque agent administre un service d'un composant. Dans le cadre de l'architecture présentée en section 2, OCE opère dans un cycle, appelé *cycle moteur*. Les agents eux-mêmes fonctionnent de manière cyclique : dans un cycle classique, appelé *cycle agent*, un agent perçoit, puis décide et enfin agit. Plusieurs cycles agent s'effectuent ainsi dans un cycle moteur : les agents perçoivent les messages reçus, décident et effectuent les actions conformément au protocole ARSA. En fin de cycle moteur, après présentation d'un assemblage à l'utilisateur, le moteur OCE récupère le feedback de l'utilisateur pour apprendre. Il peut ensuite sonder une nouvelle fois l'environnement ambiant et démarrer un nouveau cycle avec les connaissances des agents mises à jour. Cette méthode se rapproche du raisonnement par cas [AP94] basé sur la réutilisation de cas antérieurs et de solutions apprises précédemment ayant permis de résoudre un problème semblable au problème courant.

Dans un premier temps, nous présentons ce que fait un agent dans un cycle agent, en particulier comment il exploite ses connaissances. Puis, nous expliquons comment chaque agent construit et fait évoluer ses connaissances par apprentissage, et nous discutons ce mode d'apprentissage.

4.1 Cycle de vie d'un agent : de la perception à l'action

Pour décider de l'action à effectuer, un agent construit une représentation de la situation courante (4.1.1) dans la phase de perception. Il compare ensuite cette situation à des situations de référence qu'il a déjà rencontrées (4.1.2) pour pouvoir l'évaluer (4.1.3)

et choisir l'agent (et donc l'action) à qui il répond (4.1.4), dans la phase de décision. Enfin, il effectue l'action (4.1.5).

4.1.1 Construction de la situation courante

On appelle *situation courante* S_t^i pour un agent A^i la situation dans laquelle se trouve A^i dans le cycle moteur courant : elle est composée de l'ensemble des agents services perçus par A^i dans l'environnement et compatibles avec lui du point de vue de la composition [2]. Cette situation est construite par A^i, incrémentalement dans un cycle moteur, à partir des messages qu'il reçoit. C'est à l'étape de perception qu'est créée puis actualisée la situation courante de l'agent A^i conformément à l'algorithme 1.

La situation courante S_t^i est formée d'un ensemble des couples de la forme $(A^j, Type\_Message)$ où :

- A^j est l'identifiant de l'agent émetteur du message,
- $Type\_Message$ est le type de message envoyé dans le cadre du protocole ARSA c'est-à-dire *Advertise*, *Reply*, *Select* ou *Agree*.

Algorithme 1 Perception d'un agent A^i

1: Récupérer le lot de messages reçus
2: **pour** chaque message reçu **faire**
3: Extraire le couple $(A^j, Type\_Message)$
4: **si** service(A^i) et service(A^j) sont compatibles **alors**
5: Actualiser la situation courante : $S_t^i \leftarrow S_t^i \cup \{(A^j, Type\_Message)\}$
6: **fin si**
7: **fin pour**

La situation courante est ensuite rapprochée des situations de référence qu'a déjà rencontrées l'agent.

4.1.2 Rapprochement avec les situations de référence

On appelle *situation de référence* pour un agent A^i une situation identifiée lors d'un précédent cycle moteur. Un agent dispose ainsi d'un ensemble de situations de référence qui constituent sa mémoire, noté Ref^i. Nous verrons plus loin comment cette connaissance est construite et maintenue par apprentissage. A l'instar d'une situation courante, une situation de référence est composée d'un ensemble d'agents services perçu par l'agent dans l'environnement à un moment donné, agents compatibles avec lui du point de vue de la composition

Une situation de référence de A^i est formée d'un ensemble de couples de la forme $(A^j, Score_j^i)$, où :

- A^j est l'identifiant de l'agent émetteur du message,
- $Score_j^i$ est une valeur numérique qui représente pour A^i l'intérêt d'une connexion avec le service administré par A^j.

La phase de rapprochement a pour objectif d'identifier la situation courante parmi les situations de référence ou, à défaut, de sélectionner les situations de référence "similaires" à la situation courante. Le rapprochement s'effectue sur la base des identifiants des agents présents dans les situations, en faisant abstraction des types de messages et des scores.

Il est réalisé par la fonction *Calculer_Similarité* qui construit un sous-ensemble avec les situations de référence dont le degré de similarité avec S_t^i est supérieur à un seuil ξ [3]. Le sous-ensemble est réduit à S_t^i si cette situation est connue (elle fait déjà partie des situations de référence) ; il est vide si aucune situation similaire n'a été trouvée.

Étant donné l'ensemble Ref^i des situations de référence de l'agent A^i et un seuil ξ, la fonction $Calculer\_Similarité$ est définie de l'ensemble des situations courantes pour A^i, Sit^i, vers un ensemble de couples composés d'une situation de référence et d'un degré de similarité (1) : à la situation courante rencontrée à l'instant t, S_t^i, est associé l'ensemble des couples formés par une situation de référence SR_k^i et son degré de similarité d_k avec S_t^i, dans le cas où d_k est supérieur ou égal au seuil ξ.

$$\begin{aligned} Calculer\_Similarité : Sit^i &\rightarrow \mathcal{P}(Ref^i \times \mathbb{R}) \\ S_t^i &\mapsto \{(SR_k^i, d_k)\}_{k \leq |Ref^i|} \; et \; d_k \geq \xi \end{aligned} \tag{1}$$

La fonction *Calculer_Similarité* peut implanter un algorithme de *clustering* pour regrouper les situations de référence en classes. Ceci permet d'exécuter l'opération de rapprochement plus rapidement [4] : il suffira à l'agent de comparer la situation courante aux représentantes dans chacune des classes.

2. Deux services sont dits compatibles si l'un est un service fourni S_F et l'autre est un service requis S_R, et si S_F rend le service S_R ($S_R \subset S_F$).

3. Pour donner une idée, le degré de similarité entre deux situations peut être calculé sur la base la proportion d'agents en commun.

4. En particulier dans le cas des environnements ambiants où le nombre de situations de référence peut être important.

4.1.3 Marquage de la situation courante

La phase de marquage permet d'enrichir la situation courante à l'aide des situations de référence qui ont été sélectionnées dans la phase précédente, en attribuant un score aux agents de la situation courante. A l'issue de ce marquage, l'agent A^i pourra établir un classement préférentiel des agents A^j de la situation courante et en sélectionner un auquel répondre.

Le marquage est réalisé par la fonction *Marquer_Situation* qui attribue les scores à partir des situations retournées par la fonction *Calculer_Similarité*. Si la situation courante S_t^i est connue, la fonction *Calculer_Similarité* a renvoyé la situation de référence correspondante et les valeurs des scores sont reproduites à l'identique par la fonction *Marquer_Situation*. Sinon, la fonction *Marquer_Situation* calcule le score $Score_j^i$ de chaque agent A^j de S_t^i [5]. Si l'un des agents de la situation courante n'apparaît pas dans les situations de référence (c'est le cas lors de l'apparition d'un service nouveau dans l'environnement ambiant), un score arbitraire lui est attribué. Le choix de la valeur de ce score permet de favoriser la prise en compte de la nouveauté [6]. Ce choix appartient à l'agent A^i. Il gère ce choix et le fait varier en fonction de ce qu'il a appris sur la sensibilité à la nouveauté de l'utilisateur. Dans le cas où aucune situation de référence similaire n'a été trouvée, les scores sont initialisés avec une valeur arbitraire. Comme précédemment, le choix de cette valeur initiale appartient à l'agent A^i.

4.1.4 Choix de l'agent

Dans cette phase, l'agent A^i sélectionne un agent A^j de la situation courante (A^i répondra alors au message de A^j).

Cette opération est effectuée par la fonction *Choisir_Agent* qui prend en paramètre la situation courante marquée et renvoie l'agent qui maxime un critère d'optimisation. Pour cela, plusieurs stratégies sont possibles basées sur les scores ou le type de message (dans l'ordre *Agree*, *Select*, *Reply*, *Advertise*) ou une combinaison de ces deux critères.

L'algorithme 2 synthétise le comportement de l'agent lors de l'étape de décision.

5. Ce calcul peut être la moyenne des scores de A^j dans les situations de référence sélectionnées, pondérée par les degrés de similarité.

6. Par exemple, en choisissant une valeur supérieure aux scores des autres agents de la situation.

Algorithme 2 Décision d'un agent A^i

1: $Sit\_Similaires_t^i \leftarrow$ ***Calculer_Similarité***(S_t^i)
2: $Situation\_Marquée_t^i \leftarrow$ ***Marquer_Situation***$(S_t^i, Sit\_Similaires_t^i)$
3: $A^j \leftarrow$ ***Choisir_Agent***$(Situation\_Marquée_t^i)$

4.1.5 Réalisation de l'action

Dans cette étape, l'agent A^i donne suite au message de l'agent A^j choisi à l'étape précédente. Le type de message à envoyer suit, dans le cadre du protocole ARSA, celui de A^j. Ainsi, un agent continue à coopérer avec les autres agents jusqu'au traitement d'un message de type *Agree*. Dans ce cas, il accepte la connexion avec le service géré par A^j et se met en attente d'un feedback sur sa décision. L'algorithme 3 décrit le comportement de A^i dans la phase d'action.

Algorithme 3 Comportement d'un agent service

1: **selon** ($Type\_Message$)
2: **cas** *Advertize* **:**
3: envoyer un message *Reply* à A^j ;
4: **cas** *Reply* **:**
5: envoyer un message *Select* à A^j ;
6: **cas** *Select* **:**
7: envoyer un message *Agree* à A^j ;
8: **cas** *Agree* **:**
9: réaliser la connexion avec A^j ;
10: se mettre en attente d'un feedback ;
11: **fin selon**

4.2 Apprentissage à base de feedback utilisateur

Au cours des différents cycles agent (perception-décision-action), l'agent exploite ses connaissances. L'apprentissage de ces dernières s'effectue hors cycle agent, lorsque le cycle moteur arrive à son terme et que le feedback de l'utilisateur est renvoyé au moteur OCE. Ce feedback porte sur l'ensemble de l'assemblage proposé. Il est répercuté sur les agents qui composent l'assemblage. Chacun de ces agents A^i construit alors une nouvelle situation de référence à partir de sa situation courante marquée. Pour cela, A^i calcule une valeur de renforcement du score de chaque agent A^k de la situation courante, notée $r_{A^k}^i$. Le calcul des $r_{A^k}^i$ utilise une variable $\beta > 0$, dont le choix appartient à A^i.

Trois cas sont possibles :

1. *L'utilisateur a accepté l'assemblage dans son intégralité.* La décision prise par tout agent A^i de l'assemblage de choisir son partenaire A^j est bonifiée :
   $r^i_{A^j} = \beta$
   $r^i_{A^k} = 0, \forall A^k \in S^i_t$ t.q. $k \neq j$.
2. *L'utilisateur a rejeté l'assemblage dans son intégralité.* La décision prise par tout agent A^i de l'assemblage de choisir A^j est pénalisée :
   $r^i_{A^j} = -\beta$
   $r^i_{A^k} = 0, \forall A^k \in S^i_t$ t.q. $k \neq j$.
3. *L'utilisateur a modifié l'assemblage.* S'il a déconnecté (manuellement, en utilisant l'éditeur à sa disposition, cf. section 3.3) le service géré par un agent A^i du service géré par un agent A^j et reconnecté le premier au service géré par un agent A^h, on bonifie cette connexion et on pénalise celle proposée par le moteur :
   $r^i_{A^h} = (Score^i_j - Score^i_h) + \beta$
   $r^i_{A^j} = -\beta$
   $r^i_{A^k} = 0, \forall A^k \in S^i_t$ t.q. $k \neq h$ et $k \neq j$.
   Les autres connexions de l'assemblage proposé sont traitées comme une acceptation (cas 1) si elles ne sont pas modifiées ou comme un rejet (cas 2) si elle sont supprimées.

Pour calculer le score $Score^i_k$ des agents A^k figurant dans la situation courante S^i_t, l'agent A^i utilise la formule (2), inspirée de celle des algorithmes de bandit, dans laquelle $Score^i_k$ est la valeur du score de l'agent A^k et $\alpha \in [0, 1]$ est le facteur d'apprentissage :

$$Score^i_k = Score^i_k + \alpha(r^i_{A^k} - Score^i_k) \qquad (2)$$

Dans le contexte dynamique et d'imprévisibilité de notre problème, il est difficile pour un agent de se projeter sur les prochaines situations de connexion et donc de déterminer l'importance des récompenses futures. Aussi, à ce stade de notre travail, nous faisons l'hypothèse que l'action choisie n'a pas d'effet sur les récompenses futures. Dans le cas contraire, il faudrait revenir à la formule plus complète du *Q-Learning* avec un facteur d'actualisation non nul.

Dans la formule, $(1 - \alpha)Score^i_k$ représente la part d'information que l'agent A^i garde de son expérience passée et $\alpha\ r^i_{A^k}$ celle qu'il apprend dans le cycle courant. On peut noter que le score des agents non retenus dans l'assemblage diminue systématiquement. Pour l'agent sélectionné A^j, en fonction de la valeur de β, le score peut aussi diminuer mais dans une moindre proportion ; la position de A^j est donc renforcée par rapport aux agents non retenus.

Une fois la situation de référence construite, l'agent A^i la stocke dans sa base de connaissances, Ref^i. Dans le cas où cette situation est déjà dans Ref^i (cas où la situation courante correspondait à une situation déjà connue), A^i se contente de mettre à jour les scores.

4.3 Discussion

Le but de l'apprentissage d'OCE est de maximiser la satisfaction de l'utilisateur. Dans notre solution, les critères de satisfaction n'ont pas à être définis explicitement : OCE se base sur le feedback de l'utilisateur (qui accepte, refuse ou modifie un assemblage) et non sur l'évaluation de critères de qualité prédéfinis. Ce type de solution confère à OCE un caractère générique (quel que soit l'utilisateur) et évolutif (l'utilisateur peut évoluer et ses critères de satisfaction aussi).

L'apprentissage ne porte pas sur l'algorithme de décision d'un agent (hors la part d'exploration inhérente à l'apprentissage par renforcement, un agent cherche toujours à se connecter avec le "meilleur" agent). Il porte sur la construction et l'adaptation des connaissances qui amènent l'agent à améliorer ses propositions. Ainsi, l'assemblage proposé par OCE pour une situation déjà rencontrée pourra différer de celui qu'il avait proposé auparavant parce qu'entre temps il a appris certaines préférences de l'utilisateur.

Dans le processus de composition d'un assemblage, les données remontent des agents et l'assemblage émerge des propositions locales de connexions. Ce comportement est caractéristique des systèmes multi-agents, dans lesquels la fonction du système n'est explicitement définie dans aucun des agents mais émerge de leurs interactions. De la même façon, l'apprentissage est distribué entre les agents. C'est l'agent qui apprend localement en révisant ses connaissances sur ses connexions locales, en ajoutant ou en modifiant des situations de référence voire en oubliant certaines situations (par exemple celles qui sont trop anciennes). Ces situations représentent la vision locale de l'agent sur les organisations possibles au sein du système multi-agent.

L'apprentissage est donc ici un apprentissage concurrent dans lequel chaque agent est un apprenant autonome influencé par l'environnement [Boe14]. On peut toutefois se demander si la vision purement locale est suffisante. Les agents ne pourraient-ils pas coopérer davantage et échanger, par exemple, des situations de référence pour une cohérence plus forte de leurs décisions ? Dans le même ordre d'idée, les agents qui gèrent les services d'un même composant "hôte" gagneraient probablement à se coordonner (par exemple, il peut être inutile d'annoncer un service fourni par un

composant tant que les services requis par le même composant hôte ne sont pas satisfaits). Ainsi, avec des agents qui apprendraient sur d'autres agents, on s'orienterait vers un apprentissage multi-agent [AS18]. Il faut cependant noter la contribution de l'utilisateur à la cohérence de la décision globale : il évalue et contrôle les décisions d'OCE et son feedback est distribué aux agents. Transformé en connaissance, ce feedback global cadre les décisions des agents et donne une cohérence globale à l'agrégation des décisions individuelles.

5 Travaux connexes

Fondamentalement, les composants logiciels et les services sont des unités logicielles développées et déployées dans le but d'être réutilisées et composées. L'automatisation de la composition est une problématique largement traitée dans la littérature, en particulier pour ce qui concerne les services Web.

Pour F. Morh [Mor16], le problème de la composition automatique des services se divise en deux grandes classes selon que la "structure" de la composition est connue au préalable ou non. Dans le premier cas, il s'agit de trouver à l'exécution les différents services qui vont permettre de réaliser un modèle donné (plan de composition, *workflow* de services...) en l'adaptant au mieux à la situation. Par exemple, MUSIC [RBD+09] permet une adaptation contextuelle de la composition en se basant sur un modèle : les plans sont sélectionnés au moment de l'exécution afin de maximiser un critère de qualité. Dans le deuxième cas, on crée de nouveaux services qui satisfont des pré-conditions et des post-conditions, ou des besoins explicités en amont. Dans tous les cas, la composition s'effectue en mode *top-down* (à l'inverse de notre approche *bottom-up*), à partir de modèles prédéfinis ou de buts explicites.

Dans [SVV11], les auteurs présentent un état de l'art sur la composition de services en intelligence ambiante. Les solutions présentées reposent sur la formulation (sous différentes formes) d'un but à atteindre. Dans certains cas, l'utilisateur peut être impliqué et choisir une solution de composition parmi différentes compositions possibles qui lui sont présentées, comme c'est le cas dans notre architecture. Il n'est cependant pas présenté de système de composition qui repose sur l'apprentissage et, par conséquent, de contribution de l'utilisateur à un processus d'apprentissage.

Dans [RFP17], les auteurs proposent une solution à base d'apprentissage pour l'adaptation en ligne et en continu de systèmes logiciels à composants. À partir d'un but explicite et d'un ensemble de configurations connues qui satisfont ce but, il s'agit de trouver la meilleure selon des critères extrafonctionnels et non de faire émerger une fonctionnalité comme nous le proposons. L'adaptation n'est pas programmée mais apprise par renforcement à partir d'expérimentations sur les différentes configurations possibles. L'utilisateur est sollicité pour expérimenter mais pas pour donner un feedback explicite. Pour cela, les applications sont instrumentées et c'est l'environnement d'exécution qui génère les données de feedback (du type de celles mentionnées dans le point 4 de la section 3.3).

Les travaux basés sur l'apprentissage pour automatiser la composition prennent le plus souvent la qualité de service (*QoS*) comme critère (par exemple, [KAA+19]). Dans ce cadre, Wang *et al.* proposent une composition auto-adaptative de services Web en environnement dynamique afin de maximiser la *QoS* globale de la composition offerte à l'utilisateur [WZZ+10]. Pour cela, en s'inspirant de [CXRM09, DGAV05], ils modélisent la structure d'une composition de service sous la forme d'un processus de décision markovien contenant plusieurs *workflows*, la politique optimale pour le choix du meilleur *workflow* étant basée sur un algorithme de *Q-Learning*. La *QoS* n'est actuellement pas prise en compte dans notre solution ; ce pourrait être un élément de nature à enrichir la décision, obtenu par observation ou *via* le feedback de l'utilisateur.

Pour traiter l'évolutivité, le passage à l'échelle et l'optimisation dynamique de la composition, Wang *et al.* [WWH+16] étendent le travail exposé dans [WZZ+10] et proposent un *framework* multi-agent collaboratif où les agents apprennent par renforcement en utilisant l'algorithme Q-Learning. Ici, le partage d'expérience entre les agents améliore leur efficacité et leur vitesse d'apprentissage. Dans [LSL+14], les auteurs proposent une approche collaborative basée sur un algorithme d'apprentissage par renforcement appelé "Learning automata", pour adapter la composition de services Web et maintenir une *QoS* satisfaisante de la composition. Y. Charif et N. Sabouret utilisent eux aussi un protocole de coordination entre agents pour la chorégraphie dynamique de services : un agent dialogue à l'aide de requêtes et utilise un historique des conversations en guise de mémoire [CS09]. Ces approches coopératives sont intéressantes et rejoignent nos perspectives discutées en section 4.3.

6 Conclusion

Pour construire des applications ambiantes par assemblage automatique et dynamique de services logi-

ciels, notre approche se démarque des solutions existantes en faisant émerger les applications de l'environnement ambiant en mode *bottom-up*, sans expression préalable des besoins ni modèle d'assemblage prédéfini. Afin de satisfaire au mieux l'utilisateur tout en limitant sa contribution dans un contexte de forte dynamique et d'imprévisibilité, cette émergence doit être contrôlée. Pour cela, notre "moteur de composition" apprend en ligne, par renforcement et à horizon infini. C'est un système multi-agent dans lequel chaque agent gère un service (fourni ou requis). Les agents interagissent et coopèrent dans le cadre du protocole ARSA. Ils apprennent individuellement et en concurrence : chacun construit, à partir des messages qu'il reçoit, sa représentation locale de l'état du monde qui l'entoure. Il identifie ainsi la situation courante et la rapproche de situations rencontrées par le passé, ces dernières ayant été valuées à partir de données de feedback de l'utilisateur. Le rapprochement entre situations permet à l'agent de prendre des décisions de connexion pertinentes pour le service qu'il administre. En outre, le protocole ARSA et le rapprochement de situations permettent d'intégrer des services nouveaux et inconnus qui apparaissent soudainement dans l'environnement.

L'architecture fonctionnelle de notre système et le protocole ARSA ont été implémentés. Un prototype d'éditeur pour l'interaction avec l'utilisateur a été développé par ailleurs [KTAB18]. Avant de mettre en œuvre la solution d'apprentissage, il convient de finaliser la définition des différentes fonctions (calcul de similarité, marquage de situation, choix de l'agent) et les paramètres d'apprentissage. Nous pourrons alors conduire différentes expérimentations, portant sur différents cas d'utilisation identifiés, afin d'évaluer et de calibrer la solution.

Quelques questions restent ouvertes. D'une part, le temps d'apprentissage nécessaire pour que le moteur propose à l'utilisateur des applications utiles et utilisables pourrait s'avérer important. Une solution brièvement exposée en fin de section 3.4 pourrait consister à incorporer des règles métier ou des règles d'ergonomie au niveau des agents, possiblement sous la forme de situations de référence prédéfinies, afin d'accélérer l'acquisition des connaissances. Par ailleurs, notre solution d'apprentissage se base sur un feedback utilisateur capturé en phase de présentation de l'application. D'autres sources de données d'apprentissage sont à envisager (cf. section 3.3). Dans tous les cas, un équilibre doit être respecté entre, d'un côté, la qualité et la quantité du feedback et de l'autre, la nature et la fréquence de la sollicitation de l'utilisateur. La précision de la valuation des situations à partir de ce feedback ainsi que le rapprochement de situations sont d'autres points critiques pour la qualité de la décision.

D'autre part, la formule (2), inspirée de celle du *Q-Learning*, ne prend pas en compte le facteur d'actualisation γ (cf. section 4.2). Cette question doit être étudiée plus précisément sur la base de cas concrets afin de déterminer en quoi le choix d'une "action" pourrait influer sur les futures récompenses. Si tel était le cas, nous devrions réintroduire le facteur d'actualisation.

Enfin, comme nous l'avons indiqué en section 4.3, notre apprentissage est essentiellement individuel, sans échange d'information ni coordination entre les agents apprenants. L'introduction dans le système de mécanismes adéquats devrait aussi améliorer la qualité de l'apprentissage, donc de la décision au niveau de l'ensemble des agents, et par conséquent la qualité de l'assemblage global proposé à l'utilisateur.

7 Remerciements

Ce travail est en partie financé par la région Occitanie et le programme opérationnel FEDER-FSE Midi-Pyrénées et Garonne ainsi que par l'Université Paul Sabatier dans le cadre de l'opération neOCampus.

Références

[AP94] A. Aamodt and E. Plaza. Case-based reasoning : Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1) :39–59, March 1994.

[AS18] S. Albrecht and P. Stone. Autonomous Agents Modelling Other Agents : A Comprehensive Survey and Open Problems. *Artificial Intelligence*, 258 :66–95, 2018.

[Boe14] J. Boes. *Apprentissage du contrôle de systèmes complexes par l'auto-organisation coopérative d'un système multi-agent : application à la calibration de moteurs à combustion.* PhD thesis, Université de Toulouse, UPS, 2014.

[BS03] C. Bach and D. Scapin. Adaptation of ergonomic criteria to human-virtual environments interactions. In *Proc. of Interact'03*, pages 880–883. IOS Press, 2003.

[CAJ09] D. J. Cook, J. C. Augusto, and V. R. Jakkula. Ambient intelligence : Technologies, applications, and opportunities. *Pervasive*

and Mobile Computing, 5(4) :277 – 298, 2009.

[CMB18] A. Cornuéjols, L. Miclet, and V. Barra. *Apprentissage artificiel : Deep learning, concepts et algorithmes*. Eyrolles, 3ème edition, 2018.

[Cou13] J. Coutaz. Essai sans prétention sur l'Interaction Homme-Machine et son évolution. *1024 : Bulletin de la Société Informatique de France*, (1) :15–33, 2013.

[CS09] Y. Charif and N. Sabouret. Un protocole de coordination d'agents introspectifs pour la chorégraphie dynamique de services. *Revue d'Intelligence Artificielle (RSTI-RIA)*, 23(1) :47–79, 2009.

[CXRM09] K. Chen, J. Xu, and S. Reiff-Marganiec. Markov-HTN Planning Approach to Enhance Flexibility of Automatic Web Service Composition. In *IEEE Int. Conf. on Web Services*, pages 9–16. IEEE, 2009.

[DGAV05] P. Doshi, R. Goodwin, R. Akkiraju, and K. Verma. Dynamic workflow composition : Using markov decision processes. *Int. Journal of Web Services Research (IJWSR)*, 2(1) :1–17, 2005.

[Fer99] J. Ferber. *Multi-agent systems : An introduction to distributed artificial intelligence*. Addison Wesley, 1999.

[KAA+19] M. E. Khanouche, F. Attal, Y. Amirat, A. Chibani, and M. Kerkar. Clustering-based and QoS-aware services composition algorithm for ambient intelligence. *Information Sciences*, 482 :419–439, 2019.

[KC03] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1) :41–50, Jan. 2003.

[KTAB18] M. Koussaifi, S. Trouilhet, J.-P. Arcangeli, and J.-M. Bruel. Ambient intelligence users in the loop : Towards a model-driven approach. In *Software Technologies : Applications and Foundations*, pages 558–572. Springer, 2018.

[LSL+14] G. Li, D. Song, L. Liao, F. Sun, and J. Du. Learning automata-based adaptive web services composition. In *5th IEEE Int. Conf. on Software Engineering and Service Science (ICSESS)*, pages 792–795. IEEE, 2014.

[Mor16] F. Morh. *Automated Software and Service Composition*. SpringerBriefs in Computer Science. Springer, 2016.

[RBD+09] R. Rouvoy, P. Barone, Y. Ding, F. Eliassen, S. O. Hallsteinsen, J. Lorenzo, A. Mamelli, and U. Scholz. MUSIC : middleware support for self-adaptation in ubiquitous and service-oriented environments. In *Software Engineering for Self-Adaptive Systems*, volume 5525 of *LNCS*, pages 164–182. Springer, 2009.

[RFP17] R. Rodrigues Filho and B. Porter. Defining emergent software using continuous self-assembly, perception, and learning. *ACM Trans. on Autonomous and Adaptive Systems*, 12(3) :16 :1–16 :25, October 2017.

[Sad11] F. Sadri. Ambient intelligence : A survey. *ACM Computing Surveys*, 43(4) :1–66, October 2011.

[Som16] I. Sommerville. Component-based software engineering. In *Software Engineering*, chapter 16, pages 464–489. Pearson Education, 10th edition, 2016.

[SVV11] T. G. Stavropoulos, D. Vrakas, and I. Vlahavas. A survey of service composition in ambient intelligence environments. *Artificial Intelligence Review*, 40(3) :247–270, September 2011.

[Wei91] M. Weiser. The computer for the 21st century. *Scientific American*, 265 :94–104, 1991.

[WWH+16] H. Wang, X. Wang, X. Hu, X. Zhang, and M. Gu. A multi-agent reinforcement learning approach to dynamic service composition. *Information Sciences*, 363 :96–119, 2016.

[WZZ+10] H. Wang, X. Zhou, X. Zhou, W. Liu, W. Li, and A. Bouguettaya. Adaptive service composition based on reinforcement learning. In *Proc. of the Int. Conf. on Service-Oriented Computing (ICSOC)*, pages 92–107. Springer, 2010.

[YTAA18] W. Younes, S. Trouilhet, F. Adreit, and J.-P. Arcangeli. Towards an intelligent user-oriented middleware for opportunistic composition of services in ambient spaces. In *Proc. of the 5th Workshop on Middleware and Applications for the Internet of Things (M4IoT)*, pages 25–30, New York, NY, USA, 2018. ACM.

Unsupervised Information Extraction: Regularizing Discriminative Approaches with Relation Distribution Losses

Étienne Simon and **Vincent Guigue** and **Benjamin Piwowarski**
Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6
LIP6, F-75005 Paris, France
{etienne.simon, vincent.guigue, benjamin.piwowarski}@lip6.fr

Abstract

Unsupervised relation extraction aims at extracting relations between entities in text. Previous unsupervised approaches are either generative or discriminative. In a supervised setting, discriminative approaches, such as deep neural network classifiers, have demonstrated substantial improvement. However, these models are hard to train without supervision. To overcome this limitation, we introduce two losses on the predicted relations distribution. These losses improve the performance of discriminative based models, and enable us to train deep neural networks satisfactorily, surpassing current state of the art on three different datasets.

1 Introduction

Information extraction models aim at discovering the underlying semantic structure linking entities mentioned in a text. This can be used to build knowledge bases, which are widely used in several applications such as question answering (Yih et al., 2015; Berant et al., 2013), document retrieval (Dalton et al., 2014) and logical reasoning (Socher et al., 2013).

In the relation extraction (RE) task, we are interested in discovering the semantic (binary) relation that holds between two entities mentioned in text. The end goal is to extract triplets of the form (subject, relation, object). A considerable amount of work has been conducted on supervised or weakly-supervised relation extraction (Kambhatla, 2004; Zeng et al., 2015; Lin et al., 2016), with recent state-of-the-art models using deep neural networks (NN).

Developing unsupervised relation extraction models is interesting for three reasons: they (1) do not necessitate labeled data except for validating the models; (2) can uncover new relation types; and (3) can be trained from large unlabeled datasets, and then fine-tuned for specific relations.

The first unsupervised models used a clustering (Hasegawa et al., 2004; Banko et al., 2007) or generative (Yao et al., 2011, 2012) approach. The latter, which obtained state-of-the-art performance, still makes a lot of simplifying hypotheses, such as assuming that the entities are conditionally independent between themselves given the relation. To train more expressive models, a shift to discriminative approaches was necessary. The open question then becomes how to provide a sufficient learning signal to the classifier. To the best of our knowledge, only Marcheggiani and Titov (2016) followed this path by leveraging representation learning for modeling knowledge bases, and proposed to use an auto-encoder model: their encoder extracts the relation from a sentence, that the decoder uses to predict a missing entity. However, their encoder is still limited compared to its supervised counterpart (e.g. Zeng et al. (2015)) and relies on hand-crafted features extracted by natural language processing tools, containing errors and unable to discover new patterns, which might hinder performances.

More importantly, our initial experiments showed that the above model was unstable, especially when using a deep NN relation classifier. It converged to either of the two following regimes, depending on hyper-parameter settings: always predicting the same relation, or predicting a uniform distribution. To overcome these limitations, we propose to use two new losses alongside a link prediction loss based on a fill-in-the-blank task, and show experimentally that this is key to learning deep neural network models. Our contributions are the following: (i) We propose two RelDist losses: a skewness loss, which encourages the classifier to predict a class with confidence for a single sentence, and a uniformity loss, which encourages the classifier to scatter a set of sentences into different classes; (ii) We perform extensive experiments on the usual NYT+FB dataset, as well

as two new datasets; (iii) We show that our RelDist losses allow us to train a deep PCNN classifier (Zeng et al., 2015) as well as improve performance of feature-based models (Marcheggiani and Titov, 2016).

In the following, we first discuss related works (Section 2) before describing our model (Section 3) and presenting experimental results (Section 4).

2 Related work

Relation extraction is a standard language classification task: given a sentence containing two entities, the goal is to predict what is the relation linking these two entities. Most relation extraction systems need to be trained on a labeled dataset. However human annotation is expensive, and virtually impractical when a large number of relations is involved.

As a result, most systems are trained on datasets built through distant supervision (Mintz et al., 2009), a compromise between the supervised and unsupervised settings. It makes the following assumption: if a sentence contains two entities linked in a knowledge base, this sentence necessarily conveys that relation. For example, distant supervision aligns the sentence "*Hubel*$_{e_1}$ *received the Nobel Prize*$_{e_2}$ *for his discovery*" with the triplet (Hubel, award received, Nobel Prize), thus supervising the sentence with the label "award received". The resulting alignment are of a poorer quality, and even though this method can leverage large amounts of unlabeled text, the relation ontology is still fixed by a knowledge base, the resulting model being unable to discover new relations.

In the supervised setting, neural network models have demonstrated substantial improvement over approaches using hand-crafted features. In particular, piecewise convolutional neural networks (PCNN, Zeng et al., 2015) are now widely used as a basis for other improvements, such as the instance-level selective attention mechanism of Lin et al. (2016) which follows the multi-instance multi-label framework (Hoffmann et al., 2011; Surdeanu et al., 2012). The recent NN approaches however need large amount of data to achieve good performances.

In the unsupervised setting, models have no access to annotated sentences or to a knowledge base: other regularity hypotheses have to be made. The resulting models can be categorized into either the generative/clustering or discriminative approaches. The former try to cluster regularities in the text surrounding two entities, while the latter use discriminative models but have to make further hypotheses, namely that a pair of given entities always share the same relation, to provide a learning signal for the classifier.

Among clustering models, one of the earliest work is from Hasegawa et al. (2004) who propose building clusters by using cosine similarity on TF-IDF vectors for the surrounding text. Later, the OpenIE approaches (Banko et al., 2007; Angeli et al., 2015) relied upon the hypothesis that the surface form of the relation conveyed by a sentence appears in the path between the two entities in its dependency tree. However, these latter works are too dependent on the raw surface form and suffer from bad generalization. In our previous example, OpenIE will extract the triplet (Hubel, received, Nobel Prize), but simply replacing "received" by "was awarded" might produce a different relation even though the semantic remains the same.

Related to these clustering approaches, the Rel-LDA models (Yao et al., 2011, 2012) use a generative model inspired by LDA to cluster sentences: each relation defines a distribution over a high-level handcrafted set of features describing the relationship between the two entities in the text (e.g. the dependency path). However, these models are limited in their expressiveness. More importantly, depending on the set of features, they might focus on features not related to the relation extraction task.

We posit that discriminative approaches can help in going further in expressiveness, especially considering recent results with neural network models. To the best of our knowledge, the only discriminative approach to unsupervised relation extraction is the variational autoencoder approach (VAE) proposed by Marcheggiani and Titov, 2016): the encoder extracts the semantic relation from hand-crafted features of the sentence (related to those of Rel-LDA), while the decoder tries to predict one of the two entities given the relation and the other entity, using a general triplet scoring function (Nickel et al., 2011). This scoring function provides a signal since it is known to predict to some extent relation triplets given their embeddings. Among the input features of the classifiers are the entities themselves, the resulting model can thus be interpreted as an autoencoder

$$\underbrace{\text{The}}_{\text{prefix}}\ \underbrace{\textbf{sol}}_{e_1}\ \underbrace{\text{was the currency of}}_{\text{infix}}\ \underbrace{\textbf{Peru}}_{e_2}\ \underbrace{\text{between 1863 and 1985.}}_{\text{suffix}}$$

Figure 1: A sentence from Wikipedia where the conveyed relation is "currency used by". We call s the sentence with the two entities removed: $s =$ (prefix, infix, suffix).

where the encoder part benefits from an additional context. The proposed loss, based on the KL divergence between the posterior distribution over relations and a uniform prior on the relation distribution, is very unstable in practice. Our proposed approaches solve this unstability, and allows us to train expressive models such as the PCNN model (Zeng et al., 2015).

3 Model description

Our model focuses on extracting the relation between two entities in textual data, and assumes that a recognition tool has identified named entities in the text. Furthermore, like most works on relation extraction, we limit ourselves to binary relations and therefore consider sentences with two tagged entities, as shown in Figure 1.

To provide a supervision signal to our relation classifier, we follow Marcheggiani and Titov (2016) and use a *fill-in-the-blank* task, i.e. "*The* $\underline{sol}_{e_1}$ *was the currency of* $\underline{\ ?\ }_{e_2}$ *between 1863 and 1985.*". To correctly fill in the blank, we could directly learn to predict the missing entity, but in this case we would not be able to learn a relation classifier. Instead, we want to first learn that this sentence expresses the semantic relation "currency used by" before using this information for a supervised task: (i) We suppose that the relation can be predicted by the text surrounding the two entities alone (see Figure 1); (ii) We then try to predict the missing entity given the predicted relation and the other entity – this gives the supervision signal. These hypotheses lead to the following formulation of the fill-in-the-blank task:

$$p(e_{-i} \mid s, e_i) = \sum_r \underbrace{p(r \mid s)}_{\text{(i) classifier}} \underbrace{p(e_{-i} \mid r, e_i)}_{\text{(ii) link predictor}} \quad (1)$$

where e_1 and e_2 are the two entities, s is the text surrounding them and r is the relation linking them. As the link predictor can consider either entity, we use e_i to designate the given entity, and $e_{-i} = \{e_1, e_2\} \setminus \{e_i\}$ the one to predict.

The relation classifier $p(r \mid s)$ and link predictor $p(e_{-i} \mid r, e_i)$ are trained jointly to reconstruct a missing entity, but the link predictor cannot access the input sentence directly. Thus, all the required information must be condensed into r, which acts as a bottleneck. We advocate that this information is the semantic relation between the two entities.

Note that Marcheggiani and Titov (2016) did not make our first independence hypothesis. Instead, their classifier is conditioned on both e_i and e_{-i}, strongly relying on the fact that r is an information bottleneck.

In the following, we first describe the relation classifier $p(r \mid s)$ in section 3.1, before introducing the link predictor $p(e_{-i} \mid r, e_i)$ in section 3.2. Arguing that the resulting model is unstable, we describe the two new RelDist losses in section 3.3.

3.1 Unsupervised Relation Classifier

Our model for $p(r \mid s)$ follows current state-of-the-art practices for supervised relation extraction by using a piecewise convolutional neural network (PCNN, Zeng et al., 2015). The input sentence can be split into three parts separated by the two entities (see Figure 1). In a PCNN, the model outputs a representation for each part of the sentence. These are then combined to make a prediction. Figure 2 shows the network architecture that we now describe.

First, each word of s is mapped to a real-valued vector. In this work, we use standard word embedding, initialized with GloVe[1] (Pennington et al., 2014), and fine-tune them during training. Based on those embeddings, a convolutional layer detects patterns in subsequences of words. Then, a max-pooling along the text length combines all features into a fixed-size representation. Note that in our architecture, we obtained better results by using three distinct convolutions, one for each sentence part (i.e. the weights are not shared). We then apply a non-linear function ($\tanh$) and sum the three vectors into a single representation for s. Finally, this representation is fed to a softmax layer to predict the distribution over the relations. This distribution can be plugged into equation (1). Denoting

[1] 6B.50d from https://nlp.stanford.edu/projects/glove/

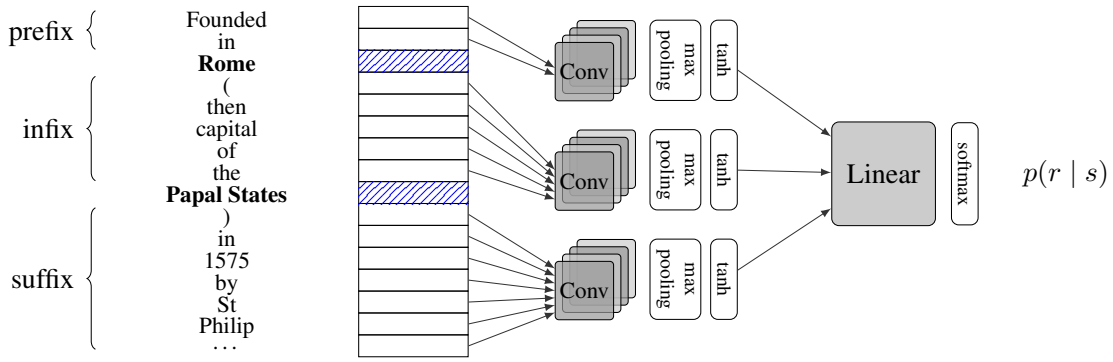


Figure 2: Our relation extraction model. Its input is the sentence with the entities removed $s = \{\text{prefix}, \text{infix}, \text{suffix}\}$. Each part is run through a convolutional layer to give a fixed-size representation, which are then fed to a softmax layer to make a prediction.

f_{PCNN} our classifier, we have:

$$p(r \mid s) = f_{\text{PCNN}}(r; s, \theta_{\text{PCNN}})$$

where θ_{PCNN} are the parameters of the classifier.

3.2 Link Predictor

The purpose of the link predictor is to provide supervision for the relation classifier. As such, it needs to be differentiable. We follow Marcheggiani and Titov (2016) to model $p(e_i \mid r, e_{-i})$, and use an energy-based formalism, where $\psi(e_1, r, e_2)$ is the energy associated with (e_1, r, e_2). The probability is obtained as follows:

$$p(e_1 \mid r, e_2) \propto \exp(\psi(e_1, r, e_2)) \tag{2}$$

where ψ is expressed as the sum of two standard relational learning models:

$$\psi(e_1, r, e_2) = \underbrace{\mathbf{u}_{e_1}^T \mathcal{A}_r \mathbf{u}_{e_2}}_{\text{RESCAL}} + \underbrace{\mathbf{u}_{e_1}^T B_r + \mathbf{u}_{e_2}^T C_r}_{\text{Selectional Preferences}}$$

where $\mathbf{u} \in \mathbb{R}^{|E| \times m}$ is an entity embedding matrix, $\mathcal{A} \in \mathbb{R}^{|R| \times m \times m}$ is a three-way tensor encoding the entities interaction and $B, C \in \mathbb{R}^{|R| \times m}$ are two matrices encoding the preferences of each relation of certain entities, and the hyper-parameter m is the dimension of the embedded entities. The function ψ also depends on the energy functions parameters $\theta_\psi = \{\mathcal{A}, B, C, \mathbf{u}\}$ that we omit for legibility. RESCAL (Nickel et al., 2011) uses a bilinear tensor product to gauge the compatibility of the two entities, whereas in the Selectional Preferences model only the predisposition of an entity to appear as the subject or object of a relation is captured.

Negative Sampling

The number of entities being very large, the partition function of equation (2) cannot be efficiently computed. To avoid the summation over the set of entities, we follow Marcheggiani and Titov (2016) and use negative sampling (Mikolov et al., 2013): instead of training a softmax classifier, we train a discriminator which tries to recognize real triplets ($D = 1$) from fake ones ($D = 0$):

$$p(D = 1 \mid e_1, e_2, r) = \sigma\left(\psi(e_1, r, e_2)\right)$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function. This model is then trained by generating negative entities for each position and optimizing the negative log likelihood:

$$\begin{aligned}\mathcal{L}_{\text{LP}} = \mathop{\mathbb{E}}_{\substack{(e_1,e_2,s)\sim\chi \\ r\sim f_{\text{PCNN}}(s)}} \Big[& -2\log\sigma\left(\psi(e_1, r, e_2)\right) \\ & - \sum_{j=1}^{k} \mathop{\mathbb{E}}_{e'\sim\mathcal{E}} \left[\log\sigma\left(-\psi(e_1, r, e')\right)\right] \\ & - \sum_{j=1}^{k} \mathop{\mathbb{E}}_{e'\sim\mathcal{E}} \left[\log\sigma\left(-\psi(e', r, e_2)\right)\right] \Big]\end{aligned} \tag{3}$$

This loss is defined over the data distribution χ, i.e. the samples (e_1, e_2, s) follow a uniform distribution over sentences containing two entities. The distribution of the relation r for the sentence s is then given by the classifier $f_{\text{PCNN}}(s)$, which corresponds to the $\sum_r p(r \mid s)$ in equation (1). Following standard practice, during training, the expectation on negative entities is approximated by sampling k random entities following the empirical entity distribution $\mathcal{E}$ for each position.

3.3 RelDist losses

Training the classifier through equation (3) alone is very unstable and dependent on precise hyper-parameter tuning. More precisely, according to our early experiments, the training process usually collapses into one of two regimes: ($\mathcal{P}1$) The classifier is very uncertain about which relation is expressed and outputs a relation following a uniform distribution ; ($\mathcal{P}2$) All sentences are classified as conveying the same relation. In both cases, the link predictor can do a good job minimizing $\mathcal{L}_{\text{LP}}$ by ignoring the output of the classifier, simply exploiting entities co-occurrences. To overcome these pitfalls, we developed two additional losses, that we now describe.

Skewness. Firstly, to encourage the classifier to be confident in its output, we minimize the entropy of the predicted relation distribution. This addresses $\mathcal{P}1$ by forcing the classifier toward outputting one-hot vectors for a given sentence using the following loss:

$$\mathcal{L}_{\text{S}} = \mathbb{E}_{(e_1,e_2,s)\sim\chi}\left[H(R \mid e_1, e_2, s)\right] \quad (4)$$

where R is the random variable corresponding to the predicted relation. Following our first independence hypothesis, the entropy of equation (4) is equivalent to $H(R \mid s)$.

Uniformity. Secondly, to ensure that the classifier predicts several relations, we minimize the KL-divergence between the prior $p(R)$ and the uniform distribution U, that is:

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}(p(R) \,\|\, U) \quad (5)$$

Note that contrary to $\mathcal{L}_{\text{S}}$, in order to have a good approximation of $p(R)$, the loss $\mathcal{L}_{\text{KL}}$ measures the un-conditionnal distribution over R, i.e. the distribution of predicted relations over all sentences. This addresses $\mathcal{P}2$ by forcing the classifier toward predicting each class equally often over a set of sentences.

To satisfactorily and jointly train the link predictor and the classifier, we use the two losses at the same time, resulting in the final loss:

$$\mathcal{L} = \mathcal{L}_{\text{LP}} + \alpha\mathcal{L}_{\text{S}} + \beta\mathcal{L}_{\text{KL}} \quad (6)$$

where α and β are both positive hyper-parameters.

All three losses are defined over the real data distribution, but in practice they are approximated at the level of a mini-batch. First, both $\mathcal{L}_{\text{LP}}$ and $\mathcal{L}_{\text{S}}$ can be computed for each sample independently. To optimize $\mathcal{L}_{\text{KL}}$ however, we need to estimate $p(R)$ at the mini-batch level, and maximize the entropy of the mean predicted relation. Formally, let s_i for $i = 1, \ldots, B$ be the i-th sentence in a batch of size B, we approximate $\mathcal{L}_{\text{KL}}$ as:

$$\sum_r \left(\sum_{i=1}^{B} \frac{f_{\text{PCNN}}(r; s_i)}{B}\right) \log \left(\sum_{i=1}^{B} \frac{f_{\text{PCNN}}(r; s_i)}{B}\right)$$

Learning We optimize the empirical estimation of (6), learning the PCNN parameters and word embeddings θ_{PCNN} as well as the link predictor parameters and entity embeddings θ_ψ jointly.

Comparison to VAE When computing the loss of the VAE model (Marcheggiani and Titov, 2016), aside from the reconstruction term $\mathcal{L}_{\text{LP}}$, the following regularization term is derived:

$$\mathcal{L}_{\text{VAEreg}} = \mathbb{E}_{(e_1,e_2,s)\sim\chi}\left[-H(R \mid e_1, e_2, s)\right]$$

This term results from the KL between $p(R \mid e_1, e_2, s)$ and the uniform distribution. Its purpose is to prevent the classifier from always predicting the same relation, i.e. it has the same purpose as our uniformity loss $\mathcal{L}_{\text{KL}}$. However its expression is equivalent to $-\mathcal{L}_{\text{S}}$, and indeed, minimizing the opposite of our skewness loss increases the entropy of the classifier output, addressing $\mathcal{P}2$. Yet, using $\mathcal{L}_{\text{VAEreg}} = -\mathcal{L}_{\text{S}}$ alone, draws the classifier into the other pitfall $\mathcal{P}1$. This causes a drop in performance, as we will show experimentally.

4 Experiments

4.1 Datasets

To evaluate our model we use labeled datasets, the labels being used for validation[2] and evaluation. The first dataset is the one of Marcheggiani and Titov (2016), which is similar to the one used in Yao et al. (2011). This dataset was built through distant supervision (Mintz et al., 2009) by aligning sentences from the New York Times corpus (NYT, Sandhaus, 2008) with Freebase (FB, Bollacker et al., 2008) triplets. Several sentences were filtered out based on features like the length of the dependency path between the two entities, resulting in 2 million sentences with only 41,000 (2%) of them labeled with one of 262 possible relations. 20% of the labeled sentences were set aside for

[2] As in other unsupervised RE papers.

validation, the remaining 80% are used to compute the final results.

We also extracted two datasets from T-REx (Elsahar et al., 2017) which was built as an alignment of Wikipedia with Wikidata (Vrandečić, 2012). We only consider triplets where both entities appear in the same sentence. We built the first dataset *DS* by extracting all triplets of T-REx where the two entities are linked by a relation in Wikidata. This is the usual distant supervision method. It resulted in 1189 relations and nearly 12 million sentences, all of them labeled with a relation.

In Wikidata, each relation is annotated with a list of associated surface forms, for example "shares border with" can be conveyed by "borders", "adjacent to", "next to", etc. The second dataset we built, *SPO*, only contains the sentence where a surface form of the relation also appears, resulting in 763,000 samples (6% of the unfiltered) and 615 relations. This dataset still contains some misalignment, but should nevertheless be easier for models to extract the correct semantic relation.

4.2 Baseline and Model

We compare our model with two state-of-the-art approaches, two generative rel-LDA models of Yao et al. (2011) and the VAE model of Marcheggiani and Titov (2016).

The two rel-LDA models only differ by the number of features considered. We use the 8 features listed in Marcheggiani and Titov (2016). Rel-LDA uses the first 3 simplest features defined in their paper, while rel-LDA1 is trained by iteratively adding more features until all 8 are used.

To assess our two main contributions individually, we evaluate the PCNN classifier and our additional losses separately. More precisely, we study the effect of the RelDist losses by looking at the differences between models optimizing $\mathcal{L}_{\text{LP}} - \alpha\mathcal{L}_{\text{S}}$ and the ones optimizing $\mathcal{L}_{\text{LP}} + \alpha\mathcal{L}_{\text{S}} + \beta\mathcal{L}_{\text{KL}}$ with $\mathcal{L}_{\text{LP}}$ being either computed using the relation classifier of Marcheggiani and Titov (2016) or our PCNN. We thus have four models: March$-\mathcal{L}_{\text{S}}$ (which corresponds to the model of Marcheggiani and Titov (2016)), March$+\mathcal{L}_{\text{S}} + \mathcal{L}_{\text{KL}}$, PCNN$-\mathcal{L}_{\text{S}}$ and PCNN$+\mathcal{L}_{\text{S}} + \mathcal{L}_{\text{KL}}$. Secondly, we study the effect of the relation classifier by comparing the feature-based classifier and the PCNN trained with the same losses.

All models are trained with 10 relation classes, which, while lower than the number of true relations, allows to compare faithfully the models since the distribution of gold relations is very unbalanced. For feature-based models, the size of the features domain range from 1 to 10 million values depending on the dataset. We train our models with Adam using L_2 regularization on all parameters. To have a good estimation of $p(R)$ in the computation of $\mathcal{L}_{\text{KL}}$, we use a batch size of 100. Words embeddings are of size 50, entities embeddings of size $m = 10$. We sample $k = 5$ negative samples to estimate $\mathcal{L}_{\text{LP}}$. Lastly, we set $\alpha = 0.01$ and $\beta = 0.02$. All three datasets come with a validation set, and following (Marcheggiani and Titov, 2016), we used it for cross-validation to optimize the B^3F_1 (described below).

4.3 Evaluation metrics

We used the B^3 metric used in Yao et al. (2011) and Marcheggiani and Titov (2016), and complemented it with two more metrics commonly seen in clustering task evaluation: V-measure (Rosenberg and Hirschberg, 2007) and ARI (Hubert and Arabie, 1985), allowing us to capture the characteristics of each approach more in detail.

To clearly describe the different metrics, we propose a common probabilistic formulation of those (in practice, they are estimated on the validation and test sets), and use the following notations. Let X (or Y) be a random variable corresponding to a sentence. We denote $c(X)$ the predicted cluster of X and $g(X)$ its conveyed gold relation.

B-cubed. The first metric we compute is a generalization of F_1 for clustering tasks called B^3 (Bagga and Baldwin, 1998). The B^3 precision and recall are defined as follows:

$$\begin{aligned}\text{B}^3\text{ Precision} &= \mathop{\mathbb{E}}_{X,Y} P\left(g(X) = g(Y) \mid c(X) = c(Y)\right)\\ \text{B}^3\text{ Recall} &= \mathop{\mathbb{E}}_{X,Y} P\left(c(X) = c(Y) \mid g(X) = g(Y)\right)\end{aligned}$$

As precision and recall can be trivially maximized by putting each sample in its own cluster or by clustering all samples into a single class, the main metric B^3 F_1 is defined as the harmonic mean of precision and recall.

V-measure. We also consider an entropy-based metric (Rosenberg and Hirschberg, 2007); this metric is defined by the homogeneity and completeness, which are akin to B^3 precision and recall, but rely on conditional entropy:

$$\begin{aligned}\text{Homogeneity} &= 1 - H\left(c(X) \mid g(X)\right)/H\left(c(X)\right)\\ \text{Completeness} &= 1 - H\left(g(X) \mid c(X)\right)/H\left(g(X)\right)\end{aligned}$$

Dataset	Model		B^3			V-measure			ARI
	Classifier	Reg.	F_1	Prec.	Rec.	F_1	Hom.	Comp.	
NYT+FB	rel-LDA		29.1	24.8	35.2	30.0	26.1	35.1	13.3
	rel-LDA1		36.9	30.4	47.0	37.4	31.9	45.1	24.2
	March.	$-\mathcal{L}_S$	35.2	23.8	67.1	27.0	18.6	49.6	18.7
	PCNN	$-\mathcal{L}_S$	27.6	24.3	31.9	24.7	21.2	29.6	15.7
	March.	$\mathcal{L}_S + \mathcal{L}_{KL}$	37.5	31.1	47.4	**38.7**	32.6	47.8	27.6
	PCNN	$\mathcal{L}_S + \mathcal{L}_{KL}$	**39.4**	32.2	50.7	38.3	32.2	47.2	**33.8**
T-REx SPO	rel-LDA		11.9	10.2	14.1	5.9	4.9	7.4	3.9
	rel-LDA1		18.5	14.3	26.1	19.4	16.1	24.5	8.6
	March.	$-\mathcal{L}_S$	24.8	20.6	31.3	23.6	19.1	30.6	12.6
	PCNN	$-\mathcal{L}_S$	25.3	19.2	37.0	23.1	18.1	31.9	10.8
	March.	$\mathcal{L}_S + \mathcal{L}_{KL}$	29.5	22.7	42.0	34.8	28.4	45.1	20.3
	PCNN	$\mathcal{L}_S + \mathcal{L}_{KL}$	**36.3**	28.4	50.3	**41.4**	33.7	53.6	**21.3**
T-REx DS	rel-LDA		9.7	6.8	17.0	8.3	6.6	11.4	2.2
	rel-LDA1		12.7	8.3	26.6	17.0	13.3	23.5	3.4
	March.	$-\mathcal{L}_S$	9.0	6.4	15.5	5.7	4.5	7.9	1.9
	PCNN	$-\mathcal{L}_S$	12.2	8.6	21.1	12.9	10.1	18.0	2.9
	March.	$\mathcal{L}_S + \mathcal{L}_{KL}$	19.5	13.3	36.7	**30.6**	24.1	42.1	**11.5**
	PCNN	$\mathcal{L}_S + \mathcal{L}_{KL}$	**19.7**	14.0	33.4	26.6	20.8	36.8	9.4

Table 1: Results (percentage) on our three datasets. The rel-LDA and rel-LDA1 models come from Yao et al. (2011). The model of Marcheggiani and Titov (2016) is March $-\mathcal{L}_S$.

As B^3, the V-measure is summarized by the F1 value. Compared to B^3, the V-measure penalizes small impurities in a relatively "pure" cluster more harshly than in less pure ones. Symmetrically, it penalizes more a degradation of a well clustered relation than of a less well clustered one.

Adjusted Rand Index. Finally, the Rand Index is defined as the probability that cluster and gold assignments are compatible:

$$\text{RI} = \mathop{\mathbb{E}}_{X,Y} \left[P\left(c(X) = c(Y) \Leftrightarrow g(X) = g(Y)\right) \right]$$

The Adjusted Rand Index (ARI, Hubert and Arabie, 1985) is a normalization of the Rand Index such that a random assignment has an ARI of 0, and the maximum is 1. Compared to the previous metrics, ARI will be less sensitive to a discrepancy between precision/homogeneity and recall/completeness since it is not an harmonic mean of both.

4.4 Results

The results reported in Table 1 are the average test scores of three runs on the NYT+FB and T-REx SPO datasets, using different random initialization of the parameters – in practice the variance was low enough so that reported results can be analyzed. We observe that regardless of the model and metrics, the highest measures are obtained on T-REx SPO, then NYT+FB and finally T-REx DS. This was to be expected, since T-REx SPO was built to be easy, and hard-to-process sentences were filtered out of NYT+FB (Yao et al., 2011; Marcheggiani and Titov, 2016). We also observe that main metrics agree in general (B^3, V-measure and ARI) in most cases. Performing a PCA on the measures, we observed that V-measure forms a nearly-orthogonal axis to B^3, and to lesser extent ARI. Hence we can focus on B^3 and V-measure in our analysis.

We first measure the benefit of our RelDist losses: on all datasets and metrics, the two models using $+\mathcal{L}_S + \mathcal{L}_{KL}$ are systematically better than the ones using $-\mathcal{L}_S$ alone: (1) The PCNN models consistently gain between 7 and 11 points in B^3 F_1 from these additional losses; (2) The feature-based classifier benefits from the RelDist losses to a lesser extent, except on the T-REx DS dataset on which the March$-\mathcal{L}_S$ model without the RelDist losses completely collapses – we hypothesize that this dataset is too hard for the model given the number of parameters to estimate.

We now restrict to discriminative models based on $+\mathcal{L}_S + \mathcal{L}_{KL}$. We note that both (March/PCNN) exhibit better performances than generative ones (Rel-LDA, Rel-LDA1) with a difference ranging from 2.5/0.6 (NYT, for March/PCNN) to 11/17.8 (on SPO). However, the advantage of PCNN over feature-based classifier is not completely clear. While the PCNN version has a systematically better B^3 F_1 on all datasets (Δ of 0.2/1.9/6.8 respectively for DS/NYT/SPO), the V-measure decreases

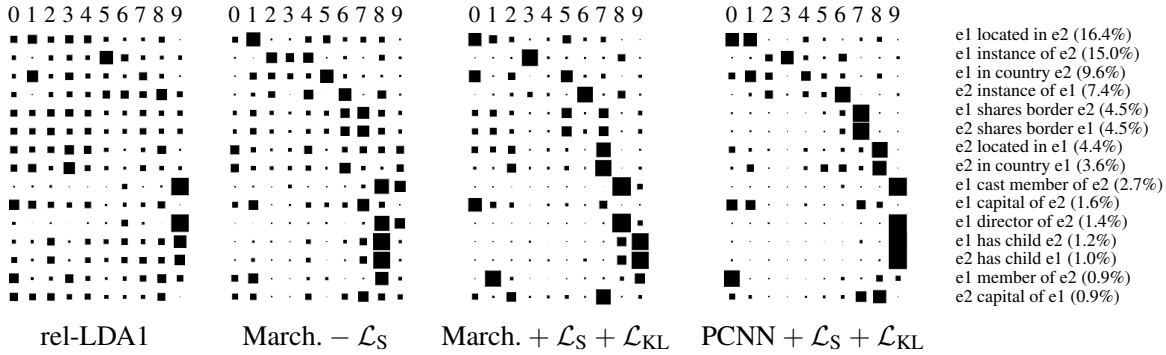


Figure 3: Normalized contingency tables for the TREx SPO dataset. Each of the 10 columns corresponds to a predicted relation cluster, which were sorted to ease comparison. The rows identify Wikidata relations sorted by frequency in the TREx SPO corpus. The area of each square is proportional to the number of sentences in the cell. The matrix was normalized so that each row sum to 1, thus it is more akin to a B^3 per-item recall than a true contingency table.

by 0.4/4.0 on respectively NYT/DS, and ARI by 2.1 on DS. As B^3 F_1 was used for validation, this shows that the PCNN models overfit this metric by polluting relatively clean clusters with unrelated sentences or degrades well clustered gold relations by splitting them within two clusters.

4.5 Qualitative Analysis

Since all the metrics agree on the SPO dataset, we plot the contingency tables of our models in Figure 3. Each row is labeled with the gold Wikidata relation extracted through distant supervision. Since relations are generally not symmetric, each Wikidata relation appears twice in the table, once for each disposition of the entities. This is particularly problematic with symmetric relations like “shares border” which are two different gold relations that actually convey the same semantic.

To interpret Figure 3, we have to see whether a predicted cluster (column) contains different gold relations – paying attention to the fact that the most important gold relations are listed in the top rows (the top 5 relations account for 50% of sentences). The first thing to notice is that the contingency tables of both models using our RelDist losses are sparser (for each columnn), which means that our models better separate relations from each other. We observe that March$-\mathcal{L}_S$ is affected by the pitfall $\mathcal{P}1$ (uniform distribution) for many gold clusters. The $-\mathcal{L}_S$ loss forces the classifier to be uncertain about which relation is expressed, translating into a dense contingency table and resulting in poor performances. The Rel-LDA1 model is even worse, and fails to identify clear clusters, showing the limitations of a purely generative approach that might focus on clusters not linked with any relation.

Focusing on our proposed model, PCNN$+\mathcal{L}_S+\mathcal{L}_{KL}$ (rightmost figure), we looked at two different mistakes. The first is a gold cluster divided in two (low recall). When looking at clusters 0 and 1, we did not find any recognizable pattern. Moreover, the corresponding link predictor parameters are very similar. This seems to be a limitation of the uniform loss: splitting a large cluster in two may improve $\mathcal{L}_{KL}$ but worsen all the evaluation metrics. The model is then penalized by the fact that it lost one slot to transmit information between the classifier and the link predictor. The second type of mistake is when a predicted cluster corresponds to two gold ones (low precision). Here, most of the mistakes seem understandable: "shares border" is symmetric (cluster 7), “located in” and “in country” (cluster 8) or “cast member” and “director of” (cluster 9) are clearly related.

5 Conclusion

In this paper, we show that discriminative RE models can be trained efficiently on unlabeled datasets by defining two losses (RelDist) on the relation distribution, that encourage the prediction to be skewed for a sentence, while being uniform for a set of sentences. In particular, we were able to successfully train a deep neural network classifier. We demonstrate the effectiveness of our RelDist losses on three datasets and showcase its effect on cluster purity.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 344–354.

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 79–85. Association for Computational Linguistics.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 365–374, New York, NY, USA. ACM.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Elena Simperl, and Frederique Laforest. 2017. T-rex: A large scale alignment of natural language with knowledge base triples. *Proceedings of the 11th International Conference on Language Resources and Evaluation*.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 415. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550. Association for Computational Linguistics.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2(1):193–218.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22. Association for Computational Linguistics.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2016. Discrete-state variational autoencoders for joint discovery and factorization of relations. *Transactions of the Association for Computational Linguistics*, 4:231–244.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

Denny Vrandečić. 2012. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st international conference on world wide web*, pages 1063–1064. ACM.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466. Association for Computational Linguistics.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 712–720. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Emnlp*, pages 1753–1762.

Apprentissage d'un modèle dynamique chaotique par un LSTM

Michele Alessandro Bucci[1], Alexandre Allauzen[1], Laurent Cordier[2], Lionel Mathelin[1], Onofrio Semeraro[1], et Guillaume Wisniewski[1]

[1]LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91 405 Orsay, France
[2]Institut Pprime, CNRS, Université de Poitiers, ENSMA, Poitiers, France

28 mai 2019

Résumé

L'objectif de ce travail est de montrer qu'il est possible d'utiliser des méthodes d'apprentissage profond pour prédire l'évolution d'un système dynamique chaotique beaucoup plus rapidement qu'en utilisant les méthodes numériques habituelles. Cette observation est une condition nécessaire au contrôle en boucle fermée d'un écoulement turbulent et soulève plusieurs problématiques nouvelles en apprentissage statistique.

Mots clés : système dynamique chaotique, prédiction de séries temporelles, LSTM

1 Introduction

Dans cet article, nous considérons la possibilité d'apprendre à prédire le comportement d'un système dynamique décrit par l'équation canonique :

$$\dot{\mathbf{v}} = f(\mathbf{v}), \tag{1}$$

où $\mathbf{v}(t) \in \mathbb{R}^n$ est le *vecteur d'état*, de dimension n, au temps t, f est une forme généralement non-linéaire de $\mathbb{R}^n$ vers $\mathbb{R}^n$ et $\dot{\mathbf{v}}$ dénote la dérivée temporelle de $\mathbf{v}$. Cette écriture, très générale, permet de décrire un très grand nombre de systèmes. Nous nous intéressons, plus particulièrement, à la modélisation d'écoulement de fluides dont le comportement est régi par les équations de Navier-Stokes qui permettent, par exemple, d'étudier la circulation du sang dans nos artères ou de simuler la trajectoire de l'air autour d'une aile d'avion ou d'une carrosserie de voiture.

Une configuration conceptuellement représentative des systèmes que nous étudions est le célèbre système dynamique de Lorenz [Lor63] qui est un modèle *minimal* de l'effet d'un gradient de température sur un fluide (convection Rayleigh-Bénard) :

$$\begin{cases} \dot{v_1} &= \sigma\,(v_2(t) - v_1(t)), \\ \dot{v_2} &= \rho\, v_1(t) - v_2(t) - v_1(t)\, v_3(t), \\ \dot{v_3} &= v_1(t)\, v_2(t) - \beta\, v_3(t), \end{cases} \tag{2}$$

où σ, ρ et β sont trois paramètres réels. [1] Le système est bien de la forme de l'Eq. (1) avec $\mathbf{v} = (v_1, v_2, v_3)$.

Le système de Lorenz fait intervenir les trois éléments caractéristiques des systèmes dynamiques : des interactions non linéaires entre les différentes composantes du vecteur d'état, des dérivées temporelles et des constantes physiques. Notons que la dimension du système, caractérisée par la dimension n de son vecteur d'état, peut varier considérablement en fonction du problème considéré. Par exemple, le système de Lorenz est de taille 3. A contrario, les simulations en mécanique des fluides ou en astro-physique impliquent usuellement des systèmes où la dimension est de l'ordre de $n \sim 10^8$ à 10^{10}.

Même s'il est régit par des équations (qui peuvent être potentiellement très simples comme en Eq. (2)), la prédiction du vecteur d'état d'un système à un instant donné fait face à deux défis. Premièrement, la présence de termes non linéaires peut donner lieu, pour certaines valeurs des paramètres physiques et/ou conditions initiales, à des comportements chaotiques (comme souvent observés dans les écoulements turbulents) dans lesquels les effets ne sont plus proportionnels aux causes et où de petites variations peuvent donc avoir des effets exponentiellement grands au cours du temps et/ou de l'espace. La figure 1 met en évidence les différences de comportement entre un système chaotique et un système non-chaotique. Deuxièmement, l'équation (1) n'admet pas, dans la plupart des cas,

1. σ représente le nombre de Prandtl,ρ joue le rôle du nombre de Rayleigh et β est un paramètre réel positif.

de solutions explicites et il est nécessaire de recourir à des simulations numériques pour observer l'évolution du système.

En dépit de son indéniable succès, [2] il existe de nombreuses situations où la simulation numérique n'est pas utilisable ou pas suffisamment fiable pour être utilisée. C'est notamment le cas lors de l'étude d'un système physique décrit par un grand nombre de degrés de liberté, typiquement les systèmes dont la description implique un large spectre d'échelles spatiales et/ou temporelles. Le contrôle en boucle fermée d'un écoulement turbulent est, de ce point de vue, un cas d'école. L'objectif du contrôle est de choisir une séquence d'actions afin de minimiser une fonction objectif (quantifiant, par exemple, la perte d'énergie ou le bruit générés par une turbulence). La prédiction de l'état de l'écoulement est alors un moyen direct de mesurer l'effet d'une action (ou d'une série d'actions) et de choisir l'action optimale. Toutefois, il faut alors que la prédiction de l'état du système contrôlé soit plus rapide que l'évolution réelle du système. Dans le cas d'un écoulement turbulent, les temps caractéristiques des phénomènes à contrôler sont très largement en-deçà de la milliseconde quand le temps de calcul est de l'ordre de la minute par pas de temps. Dans cet exemple, la simulation est ainsi de 5 à 8 ordres de grandeur plus lente que le temps réel. Ces échelles de temps sont incompatibles avec la résolution, par des méthodes numériques, des équations de Navier-Stokes, même avec les ressources de calcul d'un super-calculateur.

L'objectif de ce travail est de montrer que la prédiction du comportement d'un système dynamique chaotique soulève des problématiques nouvelles en apprentissage statistique et ouvre de nouveaux champs d'applications. Notre article est organisé de la manière suivante. Nous commençons (§2) par présenter les enjeux liés à l'application de méthodes statistiques à des systèmes dynamiques chaotiques. Puis nous montrons (§3) que des méthodes d'apprentissage profond peuvent fournir une alternative aux méthodes numériques en étant capables de prédire très rapidement (idéalement en temps réel [3]) l'état d'un système à différents horizons de temps.

2. On en fait l'expérience chaque fois que l'on monte dans un train ou une voiture !

3. Nous rappelons qu'un système est qualifié de « temps réel » lorsqu'il est capable de contrôler (ou piloter) un procédé physique à une vitesse adaptée à l'évolution du procédé contrôlé.

2 Apprentissage automatique d'un système dynamique

Principe L'idée centrale de ce travail est d'apprendre à prédire l'état d'un système dynamique (tel que défini dans l'introduction) à partir de simulations détaillées (et donc coûteuses en ressources de calcul) et/ou d'expériences physiques. Plus précisément, en faisant varier les conditions initiales ou les paramètres « physiques », nous utilisons un solveur numérique pour générer plusieurs séquences de vecteurs d'état $\left(v_t^i\right)_{t=1}^T$ où v_t^i est la valeur du vecteur d'état dans la i^{e} condition à l'instant t. Nous utilisons ensuite des méthodes d'apprentissage standard pour étudier notre capacité à *généraliser* ces observations.

Dans le contexte des systèmes dynamiques, plusieurs « types » de généralisation peuvent être envisagés :

1. étant donnée la connaissance du vecteur d'état décrivant le système à l'instant t, est-il possible de prédire l'état du système à l'instant $t+1$ ou à l'instant $t+N$ où N est un pas de temps arbitraire ?
2. étant donnée la connaissance d'une partie du système (c.-à-d. un sous-ensemble des coordonnées du vecteur d'état) est-ce que l'on est capable de prédire la totalité de celui-ci (c.-à-d. les valeurs manquantes du vecteur d'état) ?
3. étant donnée la connaissance de l'évolution pour différentes valeurs des paramètres physiques, est-ce que l'on est capable de prédire l'évolution du système si l'on change ces paramètres ?

Le premier scénario correspond au cas le plus simple dans lequel on souhaite, par exemple, pouvoir utiliser les prédictions dans un système de contrôle en boucle fermée. Le second scénario correspond à une situation plus réaliste dans laquelle seule une partie du système est observée et permettrait de prédire l'évolution d'un système « réel » dont les caractéristiques ne sont accessibles que par des capteurs placés en certains points de l'espace. Le troisième scénario, plus ambitieux, correspond à une situation dans laquelle on cherche à extrapoler le comportement d'un nouveau système à partir de l'observation de systèmes régis par les mêmes équations mais dont le comportement peut être très différent (puisque chaotique).

Dans ces trois cas, la prédiction de l'évolution du système dynamique est un problème de prédiction de série temporelle « classique » et peut être formulée comme un problème de régression qui consiste à prédire le vecteur d'état décrivant un système à l'instant $t+1$ (ou une partie de celui-ci) connaissant sa valeur (ou

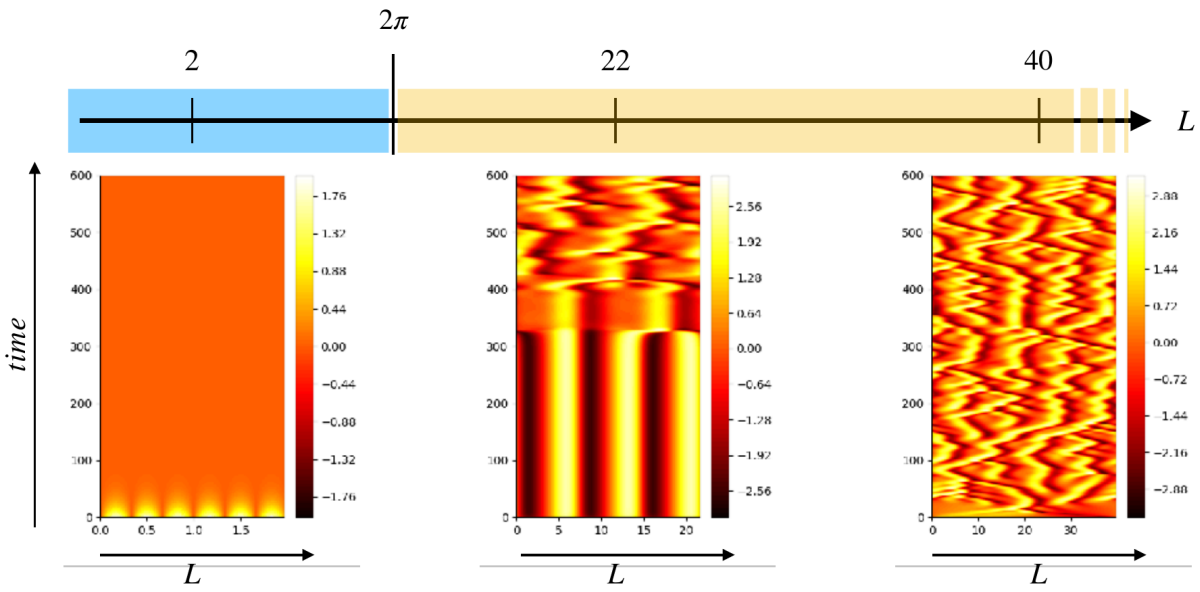


FIGURE 1 – Évolution du régime de la solution en fonction de la taille du domaine spatial L : lorsque la longueur du domaine est suffisamment petite, le système atteint une position d'équilibre (figure à gauche) ; dans le cas contraire, le système devient chaotiques (cf. les deux figures les plus à droites).

une partie de celle-ci) à l'instant t.[4] Toutefois, la possibilité de prédire le comportement d'un système potentiellement chaotique à l'aide de méthodes purement statistiques est une question fondamentale au cœur de ce travail.

Fonction de coût Comme pour tout problème d'apprentissage, la définition de la fonction coût est cruciale. La norme L_2 du résidu (*misfit*) est la définition naturelle, en bonne partie motivée par la facilité d'implémentation numérique et algorithmique de cette norme, ainsi que par l'analogie avec une énergie dans les configurations liées à la Physique. Cette fonction de coût mesure notre capacité à reproduire exactement le signal observé et ses évolutions. En particulier, elle donne la même importance à l'erreur en chaque point de l'espace.

Cependant, dans de nombreux cas concrets, il peut être suffisant de s'assurer que certaines propriétés physiques du champs sont retrouvées. Par exemple, il est fréquent de vouloir reproduire les propriétés statistiques d'une série temporelle, plutôt que de s'attacher à la reproduction exacte de la valeur à chaque instant. Dans cette optique, on privilégiera plutôt un coût impliquant les premiers moments statistiques, le spectre fréquentiel de Fourier, le spectre temps-fréquence ou la distribution des coefficients d'ondelettes ([Mal08]).

De même, il peut être plus pertinent de chercher à apprendre les propriétés de stabilité d'un système dynamique et de considérer une *perte* en termes de valeurs propres de l'opérateur d'évolution ou d'exposants de Lyapunov. Pour des champs spatiaux, dans le cas où le vecteur d'état $\mathbf{v}$ représente une quantité indexée sur des coordonnées d'espace, ou pour des distributions de probabilité, on pourrait vouloir mesurer l'erreur en termes de distance basée sur le coût du transport optimal, par exemple la distance de Wasserstein, [Vil09].

Optimiser directement ces différentes fonctions de coût lors de l'apprentissage soulève de nouveaux problèmes, notamment en termes de complexité de calculs ou d'occupation mémoire et reste un sujet ouvert.

3 Application au système de Kuramoto-Sivashinsky

Nous présentons, dans cette section, les premiers résultats que nous avons obtenus visant à prédire le comportement d'un système chaotique à l'aide d'un LSTM. Nous présenterons d'abord le système dyna-

4. Seule la manière dont les données simulées sont générées varie.

mique étudié et le modèle d'apprentissage considéré, avant de considérer deux scénarios de généralisation.

3.1 Cadre Expérimental

Nous considérons comme exemple d'application les équations de Kuramoto-Sivashinsky (KS) dans un domaine spatial uni-dimensionel périodique :

$$\frac{\partial v}{\partial t} + \sigma\, v \frac{\partial v}{\partial x} = -\rho\, \frac{\partial^2 v}{\partial x^2} - \beta\, \frac{\partial^4 v}{\partial x^4}, \tag{3}$$

où $v = v\,(x,t)$ représente le champ de vitesse et les paramètres sont fixés à $\sigma = 1$, $\rho = 1$ et $\beta = 1$. Ce modèle a initialement été développé pour étudier la dynamique des fronts de flamme en combustion.

La longueur du domaine L conditionne le régime de la solution : par exemple, si $L < L_c = 2\pi$, la solution est dynamiquement stable et converge vers une solution d'équilibre stationnaire (non-dépendante du temps). En revanche, pour $L \geq L_c$, la solution adopte un comportement chaotique (au sens de Lyapunov). Le comportement du système pour différentes valeurs de L et des conditions initiales identiques est illustré à la figure 1.

Dans ce travail, la longueur est fixée à $L = 22$ car la solution est chaotique et présente des caractéristiques similaire à celles des écoulements turbulents qui nous intéressent in fine. Une approximation de la solution de l'équation de KS est obtenue par une discrétisation en temps et en espace. Exploitant la périodicité de la solution, $v\,(x,t) = v\,(x+L,t)$, la solution est approchée à chaque instant sur une base de Fourier comportant ici 64 points de collocation. Ce niveau de discrétisation spatiale est suffisant pour une bonne approximation de la dépendance de la solution en l'espace. Un schéma semi-implicite Runge-Kutta du 3ème ordre est utilisé pour la discrétisation temporelle, avec un pas de temps $\Delta t = 0.05$.

3.2 Modèle de prédiction

Pour prédire l'évolution du système régi par les équations de Kuramoto-Sivashinsky, nous considérons un réseau LSTM [HS97] où le vecteur d'observation à chaque instant (de taille 64) constitue l'entrée de à deux couches LSTM de tailles 256. La prédiction se fait grâce à l'application d'une couche linéaire de sortie (de taille 64) également. L'apprentissage se fait de manière à minimiser l'erreur quadratique de reconstruction (ou *Mean Square Error*). L'optimisation utilise l'algorithme d'optimisation ADAM [KB15].

Prédire le vecteur d'état à l'instant $t+1$ avec un LSTM est une opération rapide qui ne nécessite que quelques opérations matricielles.

3.3 Apprentissage de séquence

Pour cette première expérience, l'objectif est de montrer qu'un réseau de neurones peut modéliser la dynamique d'un écoulement de fluide turbulent malgré le comportement chaotique de ce dernier. Les données d'apprentissage sont constituées d'une simulation sur 500 pas de temps. Après convergence, le modèle est évalué sur une séquence issue des mêmes conditions initiales.

La figure 2 représente en orange l'évolution d'une des 64 composantes du vecteur d'état prédite par le modèle LSTM. La courbe en bleue représente l'évolution temporelle de référence, simulée à partir de l'équation 3. De $t = 0$ à $t = 50$, le modèle est confronté à ses données d'apprentissage. Au delà de cette durée le modèle prédit la suite de la séquence qui n'a jamais été observée. L'apprentissage de la dynamique est quasi-parfait et ce malgré le caractère chaotique du système sous-jacent. De plus la prédiction reste bonne au-delà de $t = 50$. Ce résultat est comparable à ceux publiés dans [PHG+18] avec un modèle d'apprentissage basé sur le *reservoir computing*.

3.4 Généralisation à d'autres conditions initiales

Le modèle appris à la section 3.3 est cette fois confronté à une séquence issue du même système, mais avec des conditions initiales différentes. Le résultat est représenté à la figure 3.

Pour cela, de $t = 0$ à $t = T_0 = 25$, la mémoire du modèle est initialisée : à chaque instant t, la valeur de référence du signal est utilisée afin de faire la prédiction. Même dans cette période de mémorisation, le modèle montre un décrochage visible. Ce phénomène s'amplifie dès lors que le modèle utilise ses propres prédictions comme observations (pour $t > T_0$).

Ainsi le modèle n'est pas capable de généraliser, lorsqu'il est confronté à une séquence générée à partir de conditions initiales inconnues. Une explication serait que le modèle n'a effectivement pas été exposé lors de l'apprentissage à cette forme de diversité. Nous avons donc appris un modèle équivalent sur des séquences issues de différentes conditions initiales. Néanmoins les premières expériences montrent que le modèle peine à apprendre cette diversité et l'erreur sur les données

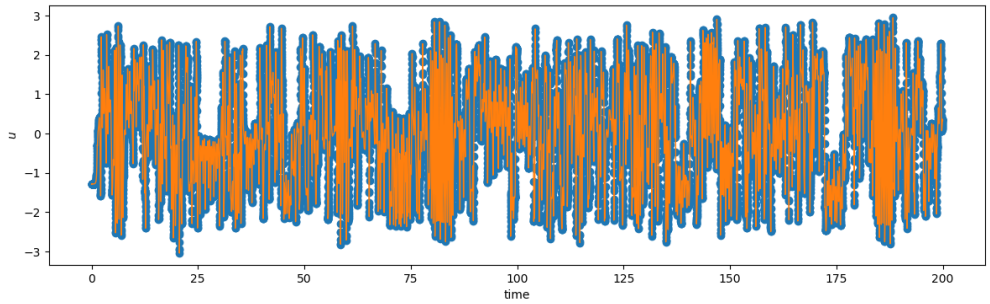


FIGURE 2 – Séquence prédite par le modèle LSTM (en orange). La courbe en bleue représente l'évolution temporelle de référence.

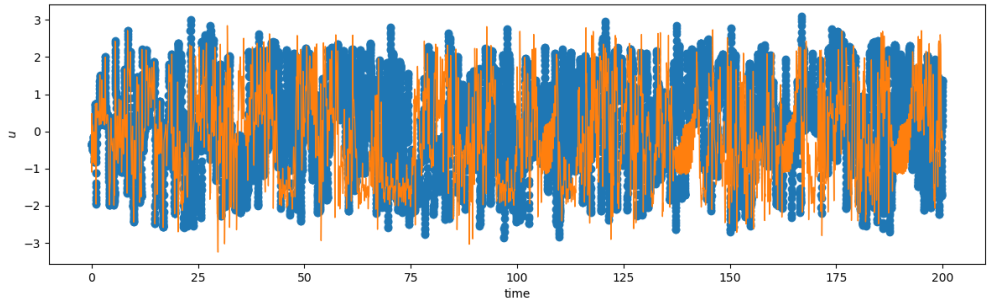


FIGURE 3 – Généralisation à d'autres conditions initiales.

d'apprentissage reste élevée. En phase de test, les prédictions ne montrent aucune amélioration.

4 Conclusion

Nous avons montré, sur un exemple précis, qu'il était possible de déterminer l'évolution d'un système dynamique chaotique à l'aide d'un LSTM, même si nous n'avons pas encore réussi à utiliser ce modèle pour prédire l'évolution d'un système dont les conditions initiales seraient différentes.

Ce premier résultat encourageant ouvre la porte à de nombreuses applications : il s'agit notamment d'une condition nécessaire pour permettre le contrôle en boucle fermée d'un système turbulent notre véritable objectif. La prochaine étape de notre travail consistera à vérifier si la qualité des prédictions est suffisante pour pouvoir utiliser une approche de type apprentissage par renforcement pour contrôler un écoulement.

5 Remerciements

Ces travaux ont été en partie financés par l'Agence Nationale de la Recherche (projet FLOWCON, ANR-17-ASTR-0022). Nous remercions les relecteurs pour leurs commentaires et suggestions.

Références

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8) :1735–1780, November 1997.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Lor63] Edward Norton Lorenz. Deterministic non-periodic flow. *Journal of the Atmospheric Sciences*, 20(2) :130–141, 1963.

[Mal08] Stéphane Mallat. *A Wavelet Tour of Signal Processing, Third Edition : The Sparse Way*. Academic Press, Inc., Orlando, FL, USA, 3rd edition, 2008.

[PHG+18] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data : a reservoir computing approach. *Physical review letters*, 120(2) :024102, 2018.

[Vil09] Cédric Villani. *Optimal Transport, Old and New*. Springer-Verlag, 2009. 976 p.

Classification des séries temporelles multivariées par l'usage de Mgrams

Angéline PLAUD *[1], Engelbert Mephu Nguifo[2], et Jacques Charreyron[3]

[1]Université Clermont Auvergne, LIMOS, CNRS et Michelin
[2]Université Clermont Auvergne, LIMOS, CNRS
[3]Michelin, Clermont-Ferrand, France

29 mai 2019

Résumé

La classification des séries temporelles univariées (STU) a déjà été explorée dans de multiples publications, où des modèles, nombreux et efficaces, ont été proposés. De tels modèles sont souvent inadéquats lorsqu'ils sont appliqués aux séries temporelles multivariées (STM). Ce type de données a émergé avec la multiplication de capteurs capables d'enregistrer de grandes quantités de données simultanément. Ces données sont caractérisées par de nombreuses dimensions, des longueurs variables, du bruit et des corrélations entre les dimensions. La classification des STM est un problème complexe et peu de travaux existent actuellement. Dans ce papier, nous proposons un nouveau modèle qui combine l'utilisation de M-histogrammes et l'apprentissage multi-vues via un ensemble de classifieurs. Le M-histogramme est un outil statistique, utilisé pour la visualisation de données, et qui peut révéler l'information utile contenue à travers les dimensions. Il peut aussi permettre l'encodage des STM. Quant à l'apprentissage multi-vues, cela correspond à l'intégration de données comme ensemble d'attributs, ce qui est le cas des STM. Ce concept peut également apporter des informations complémentaires sur les données. Notre méthode combine ces deux concepts et permet un encodage des STM qui peut être meilleur que d'autres tels que Symbolic Aggregate approXimation (SAX), au regard des expérimentations que nous avons effectuées. Nous avons comparé notre méthode avec celles de l'état de l'art sur des données de référence, et avons discuté des propriétés de notre méthode.

*Avec le soutien de Michelin et de l'ANRT

Mots-clef : Série temporelle multivariée, Mgram, Classifieur ensembliste, Apprentissage multi-vues.

1 Introduction

Une série temporelle est un ensemble de valeurs indexées par rapport au temps, représentant les mesures d'un phénomène. Lorsqu'un seul capteur est utilisé pour enregistrer ces valeurs, nous obtenons une série temporelle univariée (STU). Quand de multiples capteurs enregistrent à chaque instant des mesures, nous obtenons une série temporelle multivariée (STM). Différents domaines de recherche tels que la médecine [ODF+08], l'habitat intelligent [ZH14], l'industrie automobile [LMSZ98] se sont intéressés aux STM. L'objectif est généralement la classification de ces séries. Contrairement aux STU, les STM sont caractérisées par des interactions dans le temps mais aussi entre les dimensions. De ce fait, les méthodes de classification des STU ne sont pas applicables aux STM. WEASEL MUSE [SL17] et SMTS [BR15] font partie des modèles récents dédiés aux STM. Le premier est basé sur la transformée de Fourier, la représentation symbolique et l'extraction de fenêtres de données. Le second est basé sur les forêts aléatoires permettant la transformation d'une STM en une représentation symbolique sous forme de chaine de caractères. Ces méthodes peuvent souffrir de problèmes de scalabilité (complexité) pour les longues séries temporelles. De ce fait, la classification des STM est toujours un problème ouvert.

Dans ce papier, nous proposons un nouveau classifieur pour STM qui combine le principe d'apprentissage multi-vues [ZXXS17] avec un outil statistique appelé le M-histogramme. Celui-ci, aussi connu sous

le nom de Mgram, est un outil permettant la visualisation de la fonction de distribution inhérente aux données. Il permet la projection d'une STM dans un tableau de M dimensions. Dans notre modèle, nous l'utilisons pour réduire nos données et extraire l'information mutuelle contenue à travers plusieurs dimensions. De plus, comme chaque dimension d'une STM correspond à une vue des données, une STM peut être considérée comme une vue multiple d'un même phénomène temporel. Cette idée, de ce que nous en savons, n'a jamais été abordée dans la littérature. Cette combinaison nous donne un classifieur ensembliste qui explore les différentes vues d'une STM. Nous avons montré expérimentalement que ce modèle est efficace comparé à plusieurs autres modèles, et qu'il est robuste sur de multiples jeux de données. De plus, le Mgram fournit une nouvelle représentation des données très compétitive pour les STM par rapport à Symbolic Aggregate approXimation (SAX) [LKWL07].

Nous avons organisé le papier tel que la Section 2, est un rappel des définitions. La Section 3 est un état de l'art. La Section 4 décrit l'approche. La Section 5 présente les résultats du modèle. Finalement, la Section 6 est une conclusion.

2 Définitions

2.1 Série temporelles multivariées

Une série temporelle est un ensemble de valeurs indexées par rapport au temps, qui représente l'évolution d'un phénomène. La différence entre une série temporelle univariée (STU) et multivariée (STM) est le nombre de valeurs à chaque instant du temps (Fig.1a). Ces valeurs multiples interagissent entre elles et donc les méthodes de classification d'STU, comme reportées dans la littérature, sont inadéquates pour la classification des STM [SYWK].

Une STM X^n possède M attributs à chaque instant t durant T observations, où $T \in [1, \tau]$ et X^n est la *n-th* STM, où $n = [1, .., N]$, tel que :

$$X^n_m = [x^n_m(1), ..., x^n_m(T)] \tag{1}$$

Nous pouvons considérer chaque dimension m comme une série temporelle univariée, et donc une STM est un ensemble de STU *liées par des intéractions.* Par ailleurs, en plus des dimensions d'une STM, nous devons aussi considérer leurs longueurs. Il y a trois cas possibles :

1. *STM avec la même longueur* pour toutes les dimensions et pour toutes les STM du jeu de données. C'est le cas le plus courant de la littérature.
2. *STM avec la même longueur pour chaque STU et une longueur variable pour chaque STM.* C'est le case lorsque le même phénomène est enregistré via des mesures de même fréquence.
3. *STU et STM avec des longueurs variables.* C'est le cas lorsque le même phénomène est enregistré via des mesures de fréquences différentes et donc il y a plus ou moins de points par dimension.

Notre méthode peut gérer le premier et le deuxième cas. Le troisième cas n'est pas pris en considération dans ce travail.

2.2 Mgrams

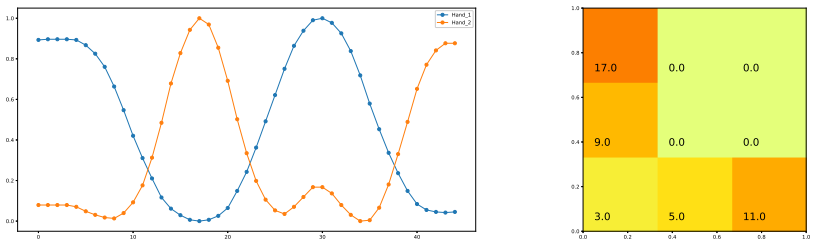
Un M-histogramme, aussi connu sous le nom de Mgram, est un outil permettant de visualiser de la fonction de densité d'un échantillon de données. Cet outil permet la projection d'une STM en un tableau de M dimensions. Les paramètres à régler pour le Mgram sont le domaine des valeurs $[min, max]$ et le nombre d'intervalles appelés *bins.* De plus, nous avons aussi le paramètre global M du Mgram. Quand $M = 1$ nous avons un 1-gram, plus connu sous le nom d'histogramme. Quand $M = 2$, nous avons un bigram. Comme le Mgram est un hypercube qui peut être difficile à representer, la Fig.1b montre un exemple de bigram.

Un bigram indique le nombre de tuples $[x^n_1(i), x^n_2(i)]$ qui sont contenus dans chaque intervalle $[b_1, b_2]$. La fréquence des tuples qui tombent dans chaque bin, permet la construction d'un tableau composé de $b_1 \times b_2$ éléments. Plus grande est la valeur d'un élément du tableau, plus grande est la fréquence de tuples contenus.

Donc la complexité d'un bigram est égale au nombre de tuples à traiter par le nombre de bins, tel que $O(T \times (B_1 + B_2))$, où B_1 et B_2 sont les nombres maximum d'intervalles sur chaque axe. Pour l'histogramme, nous avons une complexité de $O(TB)$, où B est le plus grand nombre d'intervalles.

Dans notre modèle, nous utilisons le bigram pour réduire les STM et extraire l'information mutuelle contenue dans les dimensions. L'histogramme réduit X^n à un vecteur de taille b, le nombre d'intervalles, et le bigram réduit la STM à une matrice de taille $b_1 \times b_2$. Parce que le Mgram est un outil statistique permettant d'appréhender la fonction de densité inhérente aux données, nous avons besoin de beaucoup de points pour faire une bonne estimation. Plus nous avons de poins par intervalles, plus notre Mgram est précis.

Pour éviter la perte de l'ordre des points de la STM avec cette représentation nous incluons aussi la somme



(a) Exemple d'une STM avec 2 dimensions issue du jeux de données Libras [BR15].

(b) Transformation de la STM a en un bigram avec 3 intervalles sur chaque axe.

FIGURE 1 – Exemple de transformation d'une STM en un bigram.

cumulée et la dérivée à notre modèle, en plus de la série originelle. De ce fait, le Mgram rapporte les fréquences des données de base mais aussi des dérivées et des sommes cumulées.

3 Etat de l'art

Il existe une pléthore de méthodes de classification de STU. Parmi elles, DTW-1NN [SC78], [KP01], SAX [LKWL07], [YAMP17], [MGQT13] and COTE [BLHB15] ont prouvé leur efficacité.

DTW-1NN est une méthode de classification basée sur une distance innovante DTW (Dynamic Time Warping) créée pour la reconnaissance de la parole [SC78]. Cette distance adaptée à la classification de STU, permet de calculer la similarité entre deux séries en prenant en compte les délais temporels. À l'heure actuelle, cette technique est l'une des plus connues et des plus efficaces, mais un temps de calcul élever même si des versions plus rapides ont été proposées dans la littérature [DSP+18],[PFW+16].

COTE est une méthode de classification ensembliste [Bre96], permettant à des classifieurs basiques et rapides d'être combinés. Elle utilise un système de votes sur un ensemble de distances, comme DTW, et sur un ensemble de transformations, comme shapelet [YK09]. Un shapelet étant une partie extraite d'une STU. En combinant tous les meilleurs aspects de la classification d'STU, COTE donne d'excellents résultats [BLB+17].

SAX est une méthode de transformation, permettant de passer d'une STU à une chaine de caractères. Par la même occasion, cela permet de réduire la série en gardant l'ordre des points. Cette méthode compare les nouvelles représentations grâce à une distance qui borne la distance euclidienne entre les deux séries originelles [LKWL07].

Les approches de classification de STU ne sont pas adéquates pour les STM et [SYWK] précise même que la généralisation de techniques de classification d'STU aux STM n'est pas triviale. Ceci est principalement dû à la difficulté de prendre en compte les dépendances entre les dimensions d'une STM. Cependant, il existe quand même quelques adaptations. Nous avons présenté DTW-1NN et SAX car elles font partie de ces méthodes. Par exemple, SMTS [BR15] est basé sur SAX et est une forêt aléatoire qui transforme une STM en un sac de mots. WEASEL_MUSE [SL17] transforme une STM en un histogramme/bigram comptant la fréquence de mots Symbolic Fourier Approximation (SFA) extraits tout le long de la série. Ces méthodes donnent de bons résultats, cependant leurs complexités font qu'elles ne s'appliquent pas bien aux longues séries temporelles. En effet, les forêts aléatoires ne sont pas le meilleur outil pour les gros volumes de données, de même que les techniques d'extraction de fenêtres de données.

D'autres techniques de classification de STM sont basées sur la réduction de dimensionnalité comme PCA-Eros [YS04] ou MTSC [ZC15]. En particulier, PCA [JC16](Principal Component Analysis) est la technique de réduction la plus connue. Cependant, toutes ces techniques sont très couteuses en temps. Enfin, en 2005, une autre approche utilise un ensemble d'attributs calculés à partir de la STM [KS05], comme la durée, la longueur de la série, etc.

Les meilleures méthodes, en prenant en compte l'exactitude des résultats sont WEASEL_MUSE, SMTS et différentes versions de DTW1NN, appelées DTW1NND et DTW1NNI présentées dans [SYWK].

Concernant le M-histogramme, il a été utilisé pour améliorer le contraste d'image [SKS+15]. Cette publication décrit les différents usages de l'outil pour

déterminer le seuil permettant l'amélioration du contraste d'images. Le Mgram a aussi été utilisé pour le contraste de texte [TWL02]. Finalement, depuis 1998 Michelin utilise les Mgrams pour analyser l'usage de ses pneus [LMSZ98]. Nous avons donc choisi d'appliquer les Mgrams dans le contexte de l'apprentissage multi-vu [YWW17], [ZXXS17], comme cela permet d'extraire différentes vues des données. Le multi-vue s'applique bien aux STM comme la combinaison de Mgrams permet d'avoir différentes vues de même donnée.

4 Ensemble de Mgrams

La base de nos travaux sur les STM est la notion de Mgrams. Pour simplifier l'implémentation, l'explication et l'illustration, nous nous concentrerons seulement sur les bigrams et les histogrammes dans l'application de notre modèle. L'Algorithme 1 le décrit. Le modèle est un ensemble de bigrams et d'histogrammes représentant les vues multiples de même données. Nous extrayons des dimensions au sein de la STM, créons un bigram ou un histogramme via un apprentissage, puis finalement, nous réalisons un système de vote afin d'établir une prédiction finale.

4.1 Construction d'un bigram

Comme un bigram ne se construit qu'à partir de 2 dimensions et un histogramme avec une seule, nous devons extraire ces dimensions au sein d'une STM, s'il y en a plus. Comme nous ne pouvons pas choisir, a priori, les dimensions qui donneront la meilleure classification, nous les choisissons de manière aléatoire (Alg 1 row 3.a.).

Pour la construction d'un bigram, comme montré dans la Section 2, nous avons deux paramètres à régler qui sont les bins (intervalles) b_1 et b_2 pour chaque dimension. La complexité d'apprentissage est donc définie par le nombre de combinaisons possibles de (b_1, b_2) avec répétitions car $bigram(b_1 = i, b_2 = j) \neq bigram(b_1 = j, b_2 = i)$. Soit $O(B_1 B_2)$, où B_1 et B_2 sont les nombres maximums de bins tels que $b_1 \in B_1, b_2 \in B_2$.

Ce temps d'apprentissage peut être réduit en prenant $b_1 = b_2$. Dans ce cas la complexité devient $O(2 \times B_1)$. Nous obtenons alors un bigram carré. Concernant la taille des données à stocker, la transformation permet de réduire les STM à $b_1 \times b_2$ éléments pour les bigrams et à $b_1 + b_2$ éléments pour les histogrammes, comme nous en calculons deux. Ces transformations sont paramétrées sur le jeu de données d'entrainement et appliquées au test. Puis elles sont stockées dans deux listes, qui sont ensuite utilisées afin de réaliser un système de vote de classifieurs, comme montré dans la Fig.2.

4.2 Complément d'informations

La capacité des bigrams à réduire les données est un avantage mais aussi sa principale faiblesse. Quand nous compressons les données via le bigram, nous perdons beaucoup d'informations telles que l'ordre des points. Afin de contrer ces effets, nous ajoutons deux nouvelles composantes à notre modèle qui sont, les dérivées et les sommes cumulées des données originelles.

4.2.1 Dérivée

Il est facile de trouver des cas où deux STM donnent le même bigram. Par exemple, en changeant l'ordre des points d'une STM, nous en obtenons une nouvelle, où les tendances sont différentes, mais qui donnent les mêmes bigrams. Si nous considérons que ces deux STM appartiennent à des classes différentes, le bigram de ces données n'est pas suffisant pour les différencier. C'est pourquoi nous ajoutons à notre modèle, les dérivées, telles que :

$$X'^{n}_{m} = [x^n_m(2) - x^n_m(1), ..., x^n_m(T) - x^n_m(T-1)] \quad (2)$$

Quand nous appliquons le bigram aux dérivées, nous gardons l'information liée aux tendances.

4.2.2 Somme cumulée

Le même raisonnement peut être tenu dans le cas de l'ordre d'évènements. Si nous changeons d'ordre des blocs de points au sein d'une STM, nous en obtenons une nouvelle qui donne le même bigram. Dans ce cas la dérivée n'est plus non plus toujours suffisante. C'est pourquoi nous ajoutons la somme cumulée à notre ensemble, afin de retrouver l'ordre des points. Nous la définissons telle que :

$$\chi^n_m = [x^n_m(1) + x^n_m(2), ..., \sum_{t=1}^{t=T} x^n_m(t)] \quad (3)$$

Quand nous appliquons le bigram sur la somme cumulée, nous gardons l'information à propos de l'ordre des évènements. Même si, dans notre cas, nous gardons la somme cumulée en entier, l'utilisateur peut garder seulement quelques points en considérant que l'information peut être perdue quand la somme est trop longue.

4.3 Construction finale

Dans le modèle, nous avons donc les données originelles mais aussi leurs dérivées et leurs sommes cumulées. Comme précisé précédemment, nous avons implémenté la version bigram/histogramme et nous devons donc extraire les deux dimensions de manière aléatoire. Nous réitérons ce processus plusieurs fois afin d'extraire plusieurs paires de dimensions (Alg 1 row 3). Finalement, nous réalisons un système de vote sur classifieur afin d'obtenir une prédiction finale (Alg 1 row 4). De plus, nous avons ajouté une étape d'étude de la corrélation pour supprimer les informations redondantes (Alg 1 row 2). Cela nous permet d'avoir, un vote où toute information a le même poids, et un calcul plus rapide.

Algorithm 1 Ensemble de bigrams et de histogrammes

Require: : STM, bins min max, R nombre de ré-itérations

1. Normalisation des données
2. Suppression des corrélations par rapport à un seuil défini par un utilisateur
3. k de 1 à R nombre maximum de ré-itérations
   (a) Choix aléatoire de 2 dimensions (pas déjà analysés) parmis les M attributs de la STM
   (b) Choix aléatoire de la transformation en bigram ou en histogramme
   (c) Choix aléatoire des données originelles, dérivées, sommes cumulées
   (d) Apprentissage afin de trouver le meilleur paramétrage de bins et transformation des données
4. Vote sur classifieurs multiples afin de réaliser une prédiction finale.

Le nombre de ré-itérations k est un paramètre du modèle et est défini par l'utilisateur où le nombre maximum de ré-itérations est appelé R dans le schéma Fig.2. Cette figure représente la construction du modèle final avec le concept de multi-vues. Nous pouvons voir que le paramétrage des histogrammes et bigrams est établi sur le jeu d'entrainement. Chaque bigram / histogramme permet d'établir une prédiction propre via un classifieur choisi par l'utilisateur. Finalement, nous réalisons un système de vote pour avoir une prédiction finale.

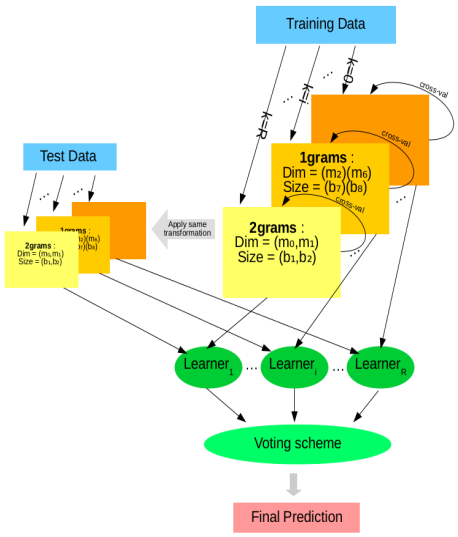


FIGURE 2 – Schéma de l'approche multi-vues constituée de bigrams et d'histogrammes.

5 Expérimentations

5.1 Conditions expérimentales

5.1.1 Les données

Le site *UEA classification* fournit 30 jeux de données à propos des STM[BDL+18]. C'est l'archive la plus fournie à l'heure actuelle, cependant la taille des séries qui sont généralement petites, ne correspond pas parfaitement à l'application des Mgrams. Par ailleurs, comme ces jeux de données ont des longueurs fixes, nous ajoutons aussi 20 jeux de données provenant du site *Baydogan* [BR15], où la taille des STM varie. Les jeux de données provenant de ce site ont déjà servi de référence aux algorithmes WEASEL_MUSE et SMTS. Par ailleurs, les deux archives présentent dans certains cas les mêmes jeux de données. Pour conclure, de tous les jeux de données, seulement *EigenWorms* correspond réellement au cas d'application des Mgrams, en terme de longueur de séries. Nous mettons à disposition une description complète des jeux de données sur notre site [PNMC19].

5.1.2 Classifieurs

Dans ce papier, l'objectif est de montrer l'intérêt des Mgrams, nous n'avons utilisé qu'un seul type de classifieur, celui du plus proche voisin (1NN).

5.1.3 Paramètres

Les paramètres à régler sont le nombre de réitérations k, le nombre de bins de chaque dimension pour le bigram b_1, b_2 et l'histogramme b. k est, dans notre implémentation, un pourcentage de combinaisons possibles lié au choix des dimensions, de la transformation et de la représentation :

$$k = pourcentage \times \binom{2}{M} \times 3 \times 2 \tag{4}$$

Pour rappel, le modèle choisit deux dimensions parmi M, puis peut effectuer deux transformations des données dérivé, somme cumulée) ou garder les données initiales, et enfin deux représentations (bigram, histogramme). Nous avons testé plusieurs pourcentages : $25\%, 50\%, 75\%$ and 95%.

Pour le nombre d'intervalles du bigram, nous avons $b_1, b_2 \in [2, 11]$ et $[b \in [2, 50]$ pour l'histogramme. Les limites sont basées sur le fait qu'ajouter plus de choix, ajoute plus d'éléments potentiellement vides dans le tableau.

Par ailleurs, dans notre cas, nous ne prenons pas en compte le paramètre de l'évolution des données, présenté dans la Section 2, car nous avons réalisé une normalisation min-max. Cela conduit les données à évoluer seulement entre $[0, 1]$. Sinon, l'utilisateur peut choisir la normalisation de son choix.

5.2 Propriétés principales

Premièrement, nous allons exposer les propriétés principales de notre modèle, montrer l'influence de ses paramètres, ainsi que démontrer sa robustesse, et sa capacité à traiter les STM de longueurs variables. De plus, nous allons décrire dans cette partie, des variantes de notre modèle telles que la version du bigram carré ou du Mgram.

5.2.1 Robustesse

Comme expliqué dans la section 5.1.3, nous avons le paramètre k fonction d'un pourcentage de combinaisons. Nous avons essayé les pourcentages $P \in [25\%, 50\%, 75\%, 95\%]$. Nous montrons ici que même si nous n'exécutons qu'une petite partie du nombre de combinaisons possibles, *i.e.* 25%, nous avons des résultats robustes.

Nous pouvons voir sur la Fig.3 que l'écart-type diverge pour assez peu de jeux de données. Dans la plupart des autres cas, les variations sont faibles. Nous pouvons donc conclure qu'une majorité des dimensions

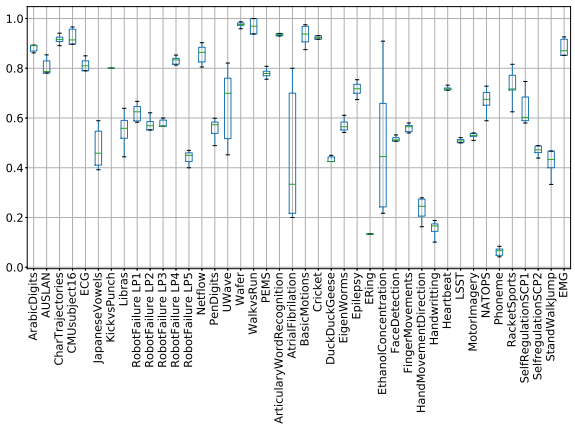


FIGURE 3 – Boîte à moustache des moyennes et des écart-types pour 25% des combinaisons.

d'une STM contiennent assez d'informations pour classifier les données. Et donc notre modèle avec 25% de combinaisons est suffisant pour émettre une classification robuste. Les autres jeux de données *AtrailFibrilation, EthanolConcentration* sont des cas spéciaux où seule une infime partie des dimensions contient réellement les informations nécessaires à la classification. De ce fait, lorsque nous extrayons les bigrams/histogrammes des mauvaises dimensions, ceux-ci ne permettent pas d'établir une bonne prédiction.

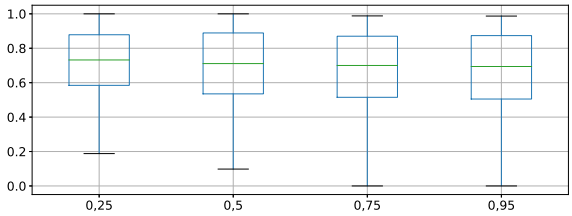


FIGURE 4 – Boîte à moustache des moyennes et écart-types pour 25%, 50%, 75%, 95% des combinaisons.

Nous montrons aussi dans la Fig.4 que le modèle ne nécessite pas le calcul de tous les bigrams/histogrammes pour avoir de bonnes performances. En effet, les moyennes sont meilleures et les écart-types sont plus petits pour 25% que pour 95%. Cela signifie qu'ajouter trop d'informations au modèle revient à ajouter du bruit dans le processus de prédiction. Dans tous les cas, il est mieux, en temps de calcul et en précision du modèle, de choisir de manière aléatoire de petits sous-ensembles de combi-

naisons plutôt que de tout calculer.

5.2.2 Longueur de séries

Dans la section 2, nous avons expliqué que les performances du Mgrams sont liées au nombre de points, et donc à la longueur, des STM. Comme le bigram est un outil statistique, le nombre de points peut impacter l'efficacité de l'outil.

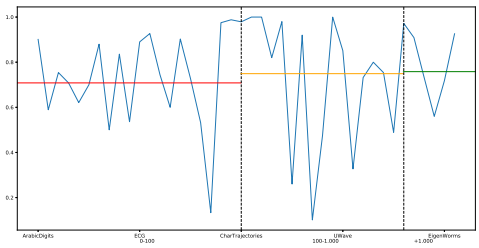


FIGURE 5 – Prédictions du modèle par taille des STM avec moyennes par intervalle de longueurs : De 0 à 100 points, de 100 à 1000 points et plus de 1000 points.

Nous pouvons voir sur la Fig.5 qu'il y a une corrélation entre la taille des STM et les performances du modèle.

5.2.3 Activités

Nous avons aussi voulu voir si notre modèle obtient de meilleures performances pour des données au sein d'un certain domaine d'activité. Nous avons donc regroupé les jeux en groupes d'activités définis par [SL17] : Ecriture, Mouvement, Capteur et reconnaissance de paroles.

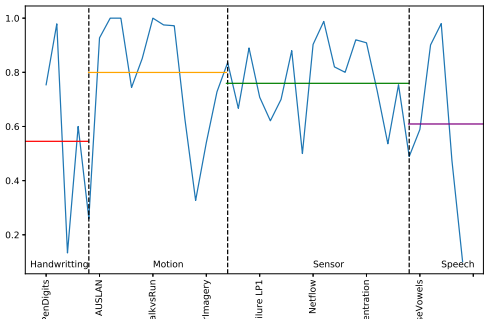


FIGURE 6 – Prédiction par domaines d'activité avec moyennes par domaines.

Nous ne pouvons pas tirer de conclusions à partir de la Fig. 6, à propos d'activités pour lesquelles notre modèle performerait mieux.

Donc nous pouvons conclure que le bigram peut être utile pour extraire, réduire, classifier les données et qu'il n'y a pas d'impact concernant les domaines d'activité.

5.3 Ensemble de Mgrams

Nous présentons ici un autre cas d'application de notre modèle.

Un Mgram est le résultat de la combinaison de toutes les dimensions M d'une STM. Nous créons alors un hypercube à M dimensions, qui compile les fréquences de tous les tuples $[x_1^n(i), x_M^n(i)]$ par bins. Nous présentons les résultats ici de ces hypercubes dans le même modèle qu'appliqué précédemment. Premièrement, il doit être précisé que cet outil ne s'adapte pas bien, sur des données possédant beaucoup de dimensions, que ce soit en temps de calcul ou en espace de stockage. Par ailleurs, les hypercubes sont difficiles à représenter dans le cadre de l'interprétation des résultats. Nous n'avons donc testé que les jeux de données où $M \leq 8$. Par exemple, si nous avons une STM avec 8 dimensions et que nous la représentons dans un tableau où chaque axe n'a que 2 bins, nous obtenons un objet qui possède 256 éléments. La taille du tableau est définie par $\prod_{i=1}^{M} b_i$ et la complexité par $O(T(\sum_{i=1}^{M} b_i))$.

Les résultats de ce modèle sont présentés dans le Tableau 1 sous l'appellation $M1gr1NN$ et mènent à deux conclusions. La première est que ce n'est pas nécessaire de calculer un Mgrams plutôt qu'un bigram. Ce modèle a une performance inférieure, en générale, à celles des bigrams comme montré sur la Fig.7. De fait, cet objet demande plus de temps de calcul et est moins pertinent qu'un bigram. La deuxième conclusion est que certaines dimensions contiennent du bruit qui délaye les performances du modèle. Ce résultat confirme ceux obtenus sur la robustesse 5.2.1 ci-avant.

5.4 Bigram carré

Durant le paramétrage des bins du bigram/histogramme, un moyen de réduire le nombre de calcul est de prendre seulement des bins identiques pour les deux dimensions, comme expliqué dans la section 4.1. Ceci représente un bon compromis dans le cas de jeux de données très larges. Nous pouvons voir dans la Fig.7 que ce modèle est toujours préférable au Mgram.

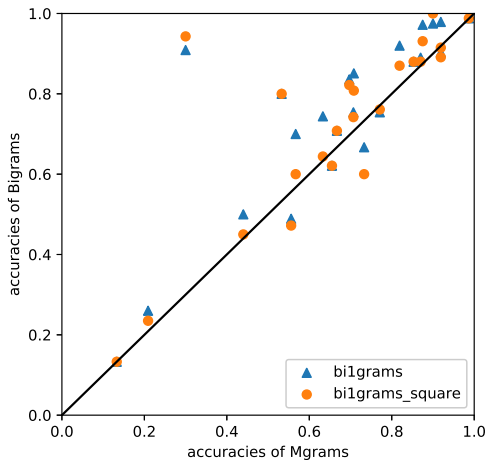


FIGURE 7 – Projection des prédictions (accuracies) des modèles bigram et bigram carré en fonction du modèle Mgram. Les points au-dessus de la droite $x = y$ signifient que les bigram et bigram carré performent mieux que le Mgram.

5.5 SAX et bigrams

Il semblait aussi important de tester si, combiner les bigrams avec d'autres méthodes de classification STU, pouvait donner des résultats intéressants. C'est pourquoi, nous avons combiné les bigrams avec SAX. Nous avons gardé notre modèle comme dans Fig.2, seulement nous avons remplacé les histogrammes par SAX. Le but est de montrer que les bigrams peuvent être combiner facilement avec d'autre méthodes de classification de STU.

5.5.1 Utilisation de SAX

L'algorithme SAX a été créé en 2007 par *Lin et al.* [LKWL07]. Pendant nos expérimentations, nous avons réglé le nombre de segments en apprentissage et nous avons fixé l'alphabet à dix chiffres de 0 à 9. Cela nous a permis de remplacer facilement les histogrammes dans le modèle initial. Nous avons choisi d'appliquer dessus une distance euclidienne.

5.5.2 Combinaison

Comme nous pouvons le voir sur le diagramme de différence critique Fig.8, le modèle Bigram-SAX performe mieux que le modèle de SAX seul. Par ailleurs, il n'y a pas de différence entre le modèle abordé précédemment et ce nouveau modèle.

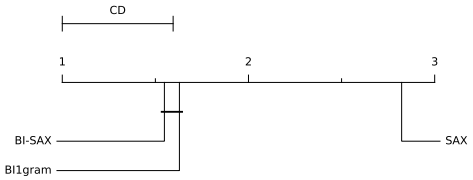


FIGURE 8 – Diagrame de différence critique pour le bigram, le modèle bigram-SAX et SAX.

Nous pouvons conclure que la combinaison de SAX avec les bigrams augmente les performances de l'algorithme seul. De ce fait, les bigrams peuvent se combiner facilement à d'autre méthode de classification et en améliorer les performances.

5.6 Prédictions sur les jeux de données de référence

Finalement, nous avons réalisé une comparaison globale avec des méthodes de la littérature.

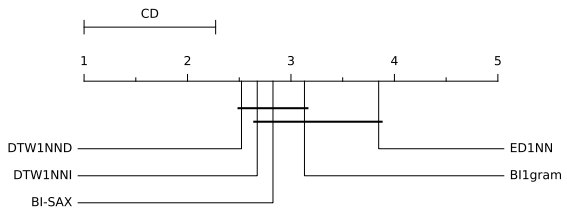
Nos méthodes, appelés $B1gr1NN$ et $BSAX1NN$ dans la table 1, sont donc comparés à $SMTS$ [BR15], $WEASEL\_MUSE$ [SL17], $DTW1NN_I$, $DTW1NN_D$ [BDL+18] comme décrits dans la Section 3. Nous avons aussi le modèle $ED1NN$ présenté dans [BDL+18], qui correspond au classifieur 1NN appliqué avec la distance euclidienne. Pour tous les algorithmes de la littérature, nous avons pris les résultats issus de leurs publications, car nous n'avions pas les paramétrages nécessaires à leurs exécutions. Les résultats pour nos modèles sont issus du meilleur paramétrage réglé par validation croisée sur le jeu d'apprentissage. Les modèles $SAX1NN$ et $BSAX1NN$ correspondent aux applications de SAX et de SAX-bigrams. Le modèle $SquarB1gr1NN$ est l'application du bigram carré quant à $M1gr1NN$ c'est le modèle Mgram.

Nous pouvons voir que nos modèles sont parfaitement compétitifs par rapport aux modèles $DTW1NN_I$, $DTW1NN_D$ dans la Fig.9a. Nous avons réalisé deux diagrammes de différence critique car les prédictions des différents modèles de la littérature ne sont pas disponibles sur les mêmes jeux de données. De plus, sur la Fig.9b nous pouvons aussi voir que bien que $SMTS$ et $WEASEL\_MUSE$ performent mieux que nos modèles il n'y a pas de différence critique. En effet, excepté le jeu de données *Japanese Vowels*, où nous avons une mauvaise performance, nos

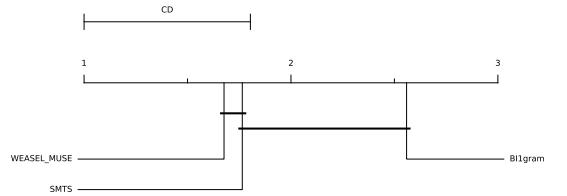
	$ED1NN$	$DTW1NN_I$	$DTW1NN_D$	$SMTS$	$WEASEL\_MUSE$	$B1gr1NN$	$SquarB1gr1NN$	$M1gr1NN$	$SAX1NN$	$BSAX1NN$
AUSLAN				0.947	**0.97**	0.927	0.902			
CharTrajectories	0.964	0.969	0.989	**0.992**	0.973	0.979	0.915	0.919		
CMUsubject16				**1.0**	**1.0**	**1.0**	**1.0**			
ECG				0.818	0.88	**0.89**	0.88	0.87		
JapaneseVowels	0.924	0.959	0.949	0.969	**0.976**	0.589	0.535			
KickvsPunch				0.82	**1.0**	**1.0**	0.8			
Libras	0.833	0.894	0.87	**0.909**	0.894	0.744	0.644	0.633	0.656	0.794
RobotFailureLP1				0.856	**0.94**	0.708	0.708	0.667	0.50	0.688
RobotFailureLP2				**0.76**	0.733	0.621	0.621	0.655	0.517	0.69
RobotFailureLP3				0.76	**0.9**	0.7	0.60	0.567	0.5	0.6
RobotFailureLP4				0.895	**0.96**	0.88	0.88	0.853	0.453	0.8
RobotFailureLP5				0.65	**0.69**	0.5	0.45	0.44	0.44	0.52
NetFlow				**0.977**	0.961	0.903	0.891	0.919		
PenDigits	0.973	0.939	**0.977**	0.917	0.912	0.754	0.742	0.707	0.785	0.818
UWave	0.881	0.868	0.903	**0.941**	0.916	0.851	0.808	0.708	0.791	0.878
Wafer				0.965	**0.997**	0.988	0.988	0.987		
WalkvsRun				**1.0**	**1.0**	**1.0**	**1.0**			
ArabicDigits	0.967	0.959	0.963	0.964	**0.992**	0.901	0.90			
ArticularyWordRecognition	0.97	0.98	**0.987**			0.98	0.96		0.677	0.98
AtrialFibrilation	0.267	0.267	0.22			**0.8**	**0.8**	0.533	0.467	**0.8**
BasicMotions	0.676	**1.0**	0.975			**1.0**	**1.0**	0.9	0.425	0.95
Cricket	0.944	0.986	**1.0**			0.972	0.931	0.875	0.472	**1.0**
DuckDuckGeese	0.275	0.55	**0.6**			0.475	**0.45**		0.2	0.2
EigenWorms	0.549	-	0.618			0.626	0	**0.718**	0.291	0.58
Epilepsy	0.666	**0.978**	0.964			0.92	0.87	0.819	0.623	0.90
EthanolConcentration	0.293	0.304	0.323			**0.943**	**0.943**	0.304	0.285	0.654
Ering	0.133	0.133	0.133			0.133	0.133	0.133	**0.3**	0.133
FaceDetection	0.519	-	0.529			**0.532**	0.511		0.503	0.517
FingerMovements	0.55	0.52	0.53			**0.6**	0.59		0.49	**0.6**
HandMovementDirection	0.278	0.306	0.231			0.33	0.33		0.365	**0.381**
HandWriting	0.2	**0.316**	0.286			0.26	0.235	0.209	0.126	0.228
Heartbeat	0.619	0.658	0.717			0.732	0.727		0.722	0.727
LSST	0.456	0.575	0.551			0.536	0.535		0.441	0.539
MotorImagery	0.51		0.5			0.56	0.56	0.440	0.5	**0.56**
NATOPS	0.85	0.85	0.883			0.728	0.733		0.594	0.767
PEMS	0.705	0.734	0.711			0.82	0.792		0.792	**0.827**
Phoneme	0.104	0.151	0.151			0.101	0.96		0.094	0.094
RacketSports	**0.868**	0.842	0.803			0.836	0.822	0.697	0.487	0.783
SelfRegulationSCP1	0.771	0.765	0.775			0.761	0.761	0.771	0.744	0.812
SelfRegulationSCP2	0.483	0.533	0.5339			0.489	0.472	0.556	544	**0.583**
StandWalkJump	0.2	0.333	0.2			**0.667**	0.6	0.733	0.467	**0.667**

TABLE 1 – Prédictions sur les jeux de données de référence.
La couleur *gris clair* correspond aux résultats que nous n'avons pas car ils n'ont pas encore été publiés. Tandis que la couleur *gris foncé* correspond aux résultats que nous n'avons pas car nous n'avons pas pu les compiler. Pour les modèles Bi-Sax et SAX, c'est car le modèle Sax est incompatible avec les variations de longueurs entre STM. Pour le modèle Mgram, c'est à cause du nombre de dimensions des STM incompatible avec le calcul de l'hypercube.

résultats sont très proches de ceux de $SMTS$ et de $WEASEL\_MUSE$. De plus, comme expliqué dans la section 4, la complexité de notre modèle est meilleure. Pour ce qui est du jeu de données *JapaneseVowels*, nous performons mal à cause de la petite taille des séries et de la multitude des dimensions. Nous pouvons conclure , que comme supposé, nous performons mieux avec des STM longues, ce qui correspond a l'usage voulu de notre modèle.



(a) Diagramme des différences critiques pour les STM avec des longueurs fixes, comparés avec $ED1NN$, $DTW1NN_D$ et $DTW1NN_I$



(b) Diagramme des différences critiques pour les STM avec des longueurs variables, comparés avec $SMTS$ et $WEASEL\_MUSE$

FIGURE 9 – Diagramme des différences critiques sur les jeux de données de référence.

6 Conclusion

L'exploitation rapide de STM de longue taille, ainsi que la représentation compacte de ces séries, sont les challenges que nous avons relevés ici. Nous avons aussi montré que les bigrams donnent de bon résultats même sans être exhaustif dans l'exploration des dimensions. Ce sont aussi des objets visuels qui permettent une compréhension et une interprétation rapide des résultats. Finalement, nous avons démontré que notre méthode a une meilleure performance sur les STM longues. Enfin, les bigrams peuvent être combinés avec d'autres méthodes STU pour généraliser leur application aux STM. Par ailleurs, notre approche étant basée sur plusieurs choix aléatoires des dimensions, il peut être envisager de rechercher d'abord des liens de dépendances entre dimensions en prélude à notre méthode.

Références

[BDL+18] A. Bagnall, H. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh. The UEA multivariate time series classification archive, 2018. *CoRR*, abs/1811.00075, 2018.

[BLB+17] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off : a review and experimental evaluation of recent algorithmic advances. *DMKD*, 31(3) :606–660, 2017.

[BLHB15] A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with cote : The collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9) :2522–2535, 2015.

[BR15] M. Baydogan and G. Runger. Learning a symbolic representation for multivariate time series classification. *DMKD*, 29(2) :400–422, 2015.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24(2) :123–140, 1996.

[DSP+18] H. Dau, D. Silva, F. Petitjean, G. Forestier, A. Bagnall, A. Mueen, and E. Keogh. Optimizing dynamic time warping's window width for time series data mining applications. *Data Mining and Knowledge Discovery*, 32(4) :1074–1120, 2018.

[JC16] I. Jolliffe and J. Cadima. Principal component analysis : a review and recent developments. *Philosophical Transactions of the Royal Society of London Series A*, 374 :20150202, 2016.

[KP01] E. Keogh and M. Pazzani. *Derivative Dynamic Time Warping*, pages 1–11. 2001.

[KS05] M. Kadous and C. Sammut. Classification of multivariate time series and structured data using constructive induction. *Machine Learning*, 58(2) :179–216, 2005.

[LKWL07] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax : a novel symbolic representation of time series. *DMKD*, 15(2) :107–144, 2007.

[LMSZ98] O. Le Maître, M. Süssner, and C. Zarak. Evaluation of tire wear performance. In *SAE Technical Paper*. SAE International, 1998.

[MGQT13] S. Malinowski, T. Guyet, R. Quiniou, and R. Tavenard. 1d-sax : A novel symbolic representation for time series. In *Advances in Intelligent Data Analysis XII*, 2013.

[ODF+08] P. Ordóñez, M. DesJardins, C. Feltes, C. U. Lehmann, and J. Fackler. Visualizing multivariate time series data to detect specific medical conditions. *Annual Symposium proceedings. AMIA Symposium*, page 530—534, 2008.

[PFW+16] F. Petitjean, G. Forestier, G. Webb, A. Nicholson, Y. Chen, and E. Keogh. Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. *Knowledge and Information Systems*, 47(1) :1–26, 2016.

[PNMC19] A. Plaud, E. NGuifo Mephu, and J. Charreyron. *MTS Mgrams*, April 2019. Available at `https://sites.google.com/view/mts-bihistograms/accueil`.

[SC78] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1) :43–49, 1978.

[SKS+15] G. Shah, A. Khan, A. Shah, M. Raza, and M. Sharif. A review on image contrast enhancement techniques using histogram equalization. *Science International*, 27 :1297–1302, 2015.

[SL17] P. Schäfer and U. Leser. Multivariate time series classification with WEASEL+MUSE. *CoRR*, abs/1711.11343, 2017.

[SYWK] M. Shokoohi-Yekta, J. Wang, and E. Keogh. *On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case*, pages 289–297.

[TWL02] C. Tan, Y. Wang, and C. Lee. The use of bigrams to enhance text categorization. *Information Processing & Management*, 38(4) :529 – 546, 2002.

[YAMP17] D. Yagoubi, R. Akbarinia, F. Masseglia, and T. Palpanas. Dpisax : Massively distributed partitioned isax. In *ICDM : International Conference on Data Mining*, pages 1–6, 2017.

[YK09] L. Ye and E. Keogh. Time series shapelets : A new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–956. ACM, 2009.

[YS04] K. Yang and C. Shahabi. A pca-based similarity measure for multivariate time series. In *2Nd ACM International Workshop on Multimedia Databases*, pages 65–74. ACM, 2004.

[YWW17] Q. Yin, S. Wu, and L. Wang. Unified subspace learning for incomplete and unlabeled multi-view data. *Pattern Recognition*, 67 :313 – 327, 2017.

[ZC15] P-Y. Zhou and K. Chan. A feature extraction method for multivariate time series classification using temporal patterns. In *Advances in Knowledge Discovery and Data Mining*, pages 409–421. Springer International Publishing, 2015.

[ZH14] J. Zbigniew and Z. Holger. The debs 2014 grand challenge. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, DEBS '14, pages 266–269. ACM, 2014.

[ZXXS17] J. Zhao, X. Xie, X. Xu, and S. Sun. Multi-view learning overview : Recent progress and new challenges. *Information Fusion*, 38 :43 – 54, 2017.

Une nouvelle approche pour rendre les réseaux de neurones robustes aux attaques malveillantes : l'entraînement antagoniste avec bruit

Alexandre Araujo[1, 2] Rafael Pinot[1, 3] Benjamin Negrevergne[1] Laurent Meunier[1, 4]
Yann Chevaleyre[1] Florian Yger[1] Jamal Atif[1]

[1]PSL, Université Paris-Dauphine [2]Wavestone [3]CEA, Université Paris-Saclay
[4]Facebook AI Research

Résumé

Depuis la découverte d'exemples malveillants capables de tromper les prédictions d'un classifieur neuronal, de nombreux chercheurs ont tenté de concevoir des techniques permettant d'entraîner des réseaux de neurones robustes à différentes attaques (en particulier contre les attaques ℓ_∞ et ℓ_2 qui sont les plus étudiées.) Cependant, on observe que les mécanismes de défense conçus pour protéger contre un type d'attaque sont souvent inefficaces contre les autres types d'attaque. Dans cet article, nous présentons une nouvelle approche d'entraînement des réseaux de neurones, intitulée RAT, pour *Randomized Adversarial Training*, ou entraînement antagoniste randomisé. Celle-ci est efficace à la fois contre les attaques ℓ_2 et ℓ_∞. Pour obtenir ce résultat, nous nous sommes appuyés sur l'apprentissage antagoniste (*adversarial learning*) classique, efficace contre les attaques ℓ_∞, que nous avons combinées avec un bruit aléatoire à l'entraînement et à l'inférence, pour améliorer les performances contre les attaques ℓ_2. Nos expériences montrent que RAT permet d'offrir une robustesse du niveau de l'état de l'art contre les attaques ℓ_2 tout en restant efficace contre les attaques ℓ_∞.

1 Introduction

Les réseaux de neurones ont récemment abouti à des performances exceptionnelles dans un large éventail de domaines tels que le traitement du langage naturel [16], la reconnaissance d'images [8] ou la reconnaissance de parole [9]. Cependant, il a été montré que ces réseaux de neurones sont vulnérables aux *attaques malveillantes* (adversarial examples en anglais), c'est à dire, des perturbations imperceptibles d'exemples naturels construites pour délibérément tromper un algorithme d'apprentissage automatique [20]. Dans la littérature, l'amplitude acceptable d'une perturbation est souvent caractérisée par les normes ℓ_2 ou ℓ_∞. Comme ces deux métriques ont des propriétés distinctes, les méthodes employées pour générer des exemples malveillants sont également différents. De manière générale, les attaques ℓ_∞ sont plus faciles à générer mais aussi plus faciles à détecter en raison de leur grande norme ℓ_2. En comparaison, les attaques malveillantes ℓ_2 nécessitent plus de puissance de calcul, mais sont plus difficiles à contrer.

L'attaque FGSM (*Fast Gradient Sign Method*) [7] et ses variantes [13] peuvent être utilisées pour générer, avec relativement peu de ressources de calcul, des exemples malveillants ℓ_∞ qui peuvent tromper les réseaux de neurones de manière quasi systématique. En même temps que leur attaque, les auteurs de [7] ont également proposé l'entraînement antagoniste (*Adversarial Training*), un premier mécanisme de défense qui consiste à ajouter des exemples malveillants au jeu de données d'entraînement pour rendre les réseaux de neurones plus robustes. L'entraînement antagoniste ℓ_∞ fonctionne bien contre les attaques ℓ_∞, mais n'offre pas de bonne protection contre les exemples malveillants générés avec d'autres types d'attaques.

En effet, depuis l'introduction de l'entraînement antagoniste, des attaques plus élaborées ont été proposées. Par exemple, [3, 14] permettent de construire des exemples malveillants avec une faible norme ℓ_2 qui mettent en échec l'entraînement antagoniste ℓ_∞, ainsi que d'autres approches de la littérature précédemment proposées. La question de généraliser l'entraînement antagoniste à d'autres types d'attaques (ℓ_2 par exemple) se pose naturellement. Cependant cela se heurte à différents verrous dont en premier lieu la faisabilité calculatoire : la génération des attaques ℓ_2 requiert une puissance de calcul considérable, et puisqu'il est nécessaire d'en générer autant que la base d'apprentissage pendant l'entraînement des modèles, cela dépasse les ressources de calcul à disposition des équipes de recherche. D'autres alternatives à l'entraînement antagoniste ont été introduites dans la littérature e.g [13], mais leur efficacité

est pour l'instant limitée, en particulier contre les attaques ℓ_2 (C&W [3]).

Dans ce papier, nous avons pour but de construire un mécanisme de défense qui est à la fois robuste aux attaques ℓ_2 et aux attaques ℓ_∞. Dans un premier temps, nous expliquons pourquoi les mécanismes de défense ℓ_∞ ne sont pas efficaces contre les attaques ℓ_2 : nous observons que les ensembles des perturbations malveillantes générées par les attaques ℓ_2 ou ℓ_∞ existantes sont strictement disjoints. Partant de cette observation, nous proposons une nouvelle méthode de défense appelée *Randomized Adversarial Training* (RAT, entraînement antagoniste randomisé) qui combine entraînement antagoniste et injection de bruit à l'inférence.

Évaluer la robustesse des réseaux de neurones randomisés est une tâche difficile. Récemment, les auteurs de [2] ont montré les limites de l'utilisation des méthodes d'évaluation classiques pour tester les techniques de défense basées sur la randomisation, et proposent d'évaluer leurs performances contre des attaques en espérance. Avec cet éclairage, nombre de défenses jusqu'alors considérées comme efficaces se sont avérées en réalité inopérantes. Nous adoptons cette technique d'évaluation (attaques en Espérance) en suivant le protocole expérimental décrit dans [2], et montrons que RAT dépasse l'état de l'art . De plus, nous évaluons toutes nos approches dans un scénario au pire cas, en considérant des attaques non ciblées (une attaque non ciblée est réussie si l'attaquant prédit n'importe qu'elle autre classe que celle initiale).

Contribution : Nous introduisons RAT, un nouveau mécanisme de défense contre les attaques malveillantes qui combine entraînement antagoniste et injection de bruit à l'entraînement et à l'inférence. Nous démontrons son efficacité contre toutes les attaques les plus puissantes actuellement. Nous menons également des expériences mettant en évidence la différence entre les attaques ℓ_2 et ℓ_∞. Enfin, nous montrons que RAT atteint l'état de l'art contre les attaques ℓ_∞ en Espérance, et bat tous les mécanismes de défense existants contre les attaques ℓ_2. En d'autres termes, RAT est à la fois plus général et plus robuste que les méthodes existantes. A notre connaissance, c'est le premier mécanisme qui montre d'aussi bons résultats sur les deux types d'attaques.

2 État de l'art

Depuis la découverte des premiers exemples malveillants dans le contexte de l'apprentissage profond par [20], de nombreux travaux de recherche ont étudié la conception d'*attaques*, permettant d'automatiser leur génération. Par exemple, l'attaque FGSM [7] (*Fast Gradient Sign Method*) est une technique qui permet de générer des exemples malveillants en calculant le gradient de la fonction de perte relatif à l'entrée x, puis, en perturbant x dans la direction du signe de ce gradient. Les auteurs dans [7] ont démontré que, pour un modèle linéaire, cette technique permettait de maximiser l'effet d'une perturbation bornée en norme ℓ_∞. Par ailleurs, leurs expériences montrent que l'attaque FGSM est également efficace contre des modèles plus complexes, tels que les réseaux de neurones.

L'approche préconisée par [7], pour se protéger contre ces attaques, est d'utiliser l'*entraînement antagoniste* (ou *Adversarial Training*), c'est-à-dire, entraîner les réseaux de neurones avec des exemples malveillants générés grâce à l'attaque FGSM.

L'attaque FGSM peut être interprétée comme une seule étape d'une descente de gradient. Pour améliorer la performance de FGSM, [11, 13] ont donc logiquement proposé des variantes de celle-ci qui effectuent *plusieurs* étapes de descente de gradient. Par exemple, [13] a proposé *Pojected Gradient Descent* (PGD), une attaque qui permet de générer des exemples malveillants, efficaces même contre les réseaux construits par entraînement antagoniste FGSM. Sans surprise, il est possible de se protéger contre les attaques PGD en utilisant des exemples malveillants générés par PGD durant l'entraînement, mais cela augmente considérablement la durée de l'entraînement.

Récemment, un certain nombre de techniques alternatives ont été proposées [19, 5, 18]. Elles montrent de bonnes performances contre différents scénarios d'attaque. Malheureusement ces techniques n'offrent pas de véritable gain de robustesse en général. Le lecteur intéressé pourra se référer à [2] pour une analyse détaillée des performances des techniques de défense.

Des attaques ont également été proposées par [14, 3], pour générer des exemples malveillants qui minimisent la norme ℓ_2 de la perturbation. Dans ce papier, nos expériences contre les attaques ℓ_2 ont principalement été conduites contre l'attaque (C&W) proposée dans [3]. Cette attaque obtient de très bons résultats contre tous types de réseaux, mais nécessite un temps de calcul prohibitif (plusieurs minutes ou plusieurs heures). Néanmoins, cette attaque reste la plus efficace, et le seul mécanisme de protection qui fonctionne contre C&W ([13]) n'offre encore qu'une protection limitée.

Une idée importante pour mieux protéger les modèles contre les attaques ℓ_2 puissantes, est d'ajouter du bruit à l'inférence. La présence de bruit complique la conception d'exemples malveillants puisque le comportement du modèle n'est pas connu à l'avance par l'attaquant. La première méthode de défense qui s'appuie explicitement sur l'injection du bruit a été présentée par [21]. Depuis, de nombreuses méthodes similaires on été proposées par [12, 17]. Malheureusement ces mécanismes de défense par ajout de bruit ne fonctionnent que contre les attaques déterministes, et leurs performances s'effondrent face à des attaques qui

calculent leur perturbation pour maximiser l'impact *en espérance* [1]. Néanmoins, les attaques en espérance sont coûteuses, et l'injection de bruit reste à ce jour la seule technique de défense un tant soit peu efficace contre ce type d'attaques.

Quelques travaux se sont intéressés à l'étude théorique de la défense par injection de bruit. On peut citer les travaux de [6] qui permettent de calculer des bornes sur la perturbation minimale nécessaire pour tromper un réseau de neurones, ou de [15] qui mettent en évidence l'intérêt de la famille exponentielle dans le processus de défense.

3 Préliminaires

Dans cette partie, nous rappelons d'abord quelques définitions nécessaires pour la suite de ce papier, puis, le principe de fonctionnement des principales attaques.

3.1 Réseaux de neurones randomisés

Dans un premier temps, nous rappelons la définition d'un réseau de neurones classique et celle d'un réseau de neurones randomisés.

Définition 1 (Réseau de neurones)**.** *Étant donné un ensemble de N fonctions $\phi^{(1)}_{\theta_1}, \ldots, \phi^{(N)}_{\theta_N}$, paramétrées par les vecteurs de paramètres $\theta_i, \ldots, \theta_N$, on définit un réseau de neurones F_Θ comme la composition des fonctions $\phi^{(i)}$ suivante :*

$$F_\Theta(x) := \phi^{(N)}_{\theta_N} \circ \cdots \circ \phi^{(1)}_{\theta_1}(x),$$

où Θ représente l'ensemble des paramètres du réseau θ_1, $\ldots$, θ_N.

Note : dans cette définition, chaque fonction $\phi^{(i)}_{\theta_i}$ représente une couche *du réseau, et peut être de différentes natures (e.g. convolutionelle, dense ou autre).*

On définit maintenant le concept de réseau de neurones randomisé de la manière suivante :

Définition 2 (Réseau de neurones bruité)**.** *Un réseau de neurones randomisé, est un réseau de neurones auquel on a ajouté un bruit gaussien b :*

$$\tilde{F}_\Theta(x) := \phi^{(N)}_{\theta_N} \circ \cdots \circ (\phi^{(1)}_{\theta_1}(x) + b).$$

Où b est un vecteur aléatoire tiré de la loi normale $\mathcal{N}(0, \sigma I)$.

D'après [15], il est possible d'utiliser n'importe quel type de bruit issu de la famille des lois exponentielles. Il est également possible d'injecter du bruit à la sortie de n'importe quelle couche. Cependant, par souci de simplicité, nous limitons notre étude aux bruits gaussiens ajoutés au résultat de la première couche.

3.2 Attaques

Les attaques existantes se distinguent avant tout par la norme utilisée pour quantifier la taille de la perturbation : ℓ_2 ou ℓ_∞. Une fois, la norme fixée, trouver un exemple malveillant est un problème d'optimisation bi-critère : il s'agit de minimiser la norme de la perturbation tout en maximisant la fonction de perte. Pour simplifier le processus d'optimisation, certaines attaques minimisent la norme de la perturbation en imposant une borne inférieure sur la perte (approche de type 1). D'autres maximisent la perte en imposant une borne supérieure sur la norme de la perturbation (approche de type 2). Dans le paragraphe qui suit, nous décrivons différents types d'attaques ℓ_∞ et ℓ_2 qui implémentent l'une ou l'autre de ces approches.

Attaque 1 (FGM : Goodfellow et al. [7], attaque ℓ_∞ de type 2)**.** *L'attaque* Fast Gradient Method with ℓ_p-norm *(FGM_p) vise à générer des exemples malveillants en résolvant le problème d'optimisation suivant :*

$$\underset{||r||_p \leq \epsilon}{\operatorname{argmax}} \mathcal{L}(F_\Theta(x + r), y)$$

, où $\mathcal{L}$ dénote la fonction de perte du classifieur (en l'occurrence, l'entropie croisée) et y représente l'étiquette de l'exemple non perturbé x. Comme ϵ est faible, il est possible d'approcher la solution du problème précédent en résolvant le problème suivant :

$$\underset{||r||_p \leq \epsilon}{\operatorname{argmax}} \nabla_x \mathcal{L}(F_\theta(x), y)^\top r$$

.

Remarque : Lorsque $p = \infty$, choisir $r = \epsilon \cdot \text{sign}(\nabla_x \mathcal{L})$, maximise l'expression ci-dessus. Cette remarque a donné lieu à l'attaque FGSM qui permet de générer efficacement des perturbations en norme ℓ_∞.

Attaque 2 (C&W : Carlini and Wagner [3], attaque ℓ_2 de type 1)**.** *Pour construire un exemple malveillant ℓ_2, les auteurs de [3] proposent de résoudre le problème d'optimisation suivant :*

$$\underset{r}{\operatorname{argmin}} ||r||_2 + c \cdot g(x + r)$$

Où $g(\cdot)$ est une fonction qui retourne une valeur négative si et seulement si le classifieur f dérivé de F_Θ prédit une étiquette différente de la vrai étiquette y de l'exemple x non perturbé, (i.e. si $f(x + p) \neq y$). Puis, ils utilisent une descente de gradient stochastique ou l'optimiseur Adam pour trouver une solution approchée à ce problème d'optimisation.

Remarque : l'attaque C&W peut être modifiée pour produire des attaques ℓ_∞, mais les résultats obtenus sont moins performants que les attaques ℓ_∞ classiques comme FGSM.

Attaque 3 (PGD : Kurakin et al. [10], Madry et al. [13], attaque ℓ_2 ou ℓ_∞ de type 2)**.** *L'attaque PGD (*Projected Gradient Descent*) est une généralisation de FGSM [10] qui, comme l'attaque FGM, s'appuie sur la résolution du problème d'optimisation suivant :*

$$\underset{||r||_p \leq \epsilon}{\operatorname{argmax}} \mathcal{L}(F_\Theta(x+r), y)$$

Pour pouvoir résoudre ce problème efficacement, les auteurs proposent une méthode itérative qui permet d'obtenir une solution perturbée après un nombre fixe d'itérations déterminé par l'utilisateur. À chaque itération, on effectue l'opération suivante pour obtenir l'exemple perturbé x^{t+1} à partir de l'exemple perturbé x^t trouvé à l'itération t :

$$x^{t+1} = \prod_{x \oplus r} (x^t + \alpha \cdot \operatorname{sign}(\nabla_x \mathcal{L}(F_\Theta(x^t), y))),$$

où $x \oplus r$ est la somme de Minkowski entre $\{x\}$ et $\{r$ t.q. $||r||_p \leq \epsilon\}$, α est la taille d'un pas de gradient, $\prod_S$ est l'opérateur de projection sur S et x^0 est choisi aléatoirement dans $x \oplus r$.

Dans le reste de ce papier on utilise PGD^t pour désigner une attaque PGD à t itérations.

Remarque concernant l'utilisation d' attaques déterministes contre des réseaux randomisés : Les attaques déterministes décrites plus haut n'obtiennent pas de bonnes performances contre les réseaux de neurones randomisés, mais il est possible de les rendre performantes en utilisant *Expectation Over Transformation* (EoT), une méthode décrite dans [1, 4] pour produire des attaques robustes à certaines variations stochastiques comme la présence de bruit.

Soit $\tilde{F}_\Theta(x)$, un réseau bruité (on note b le bruit associé) évalué sur un exemple x. L'attaque EoT correspondant calcule par descente de gradient la perturbation qui maximise l'erreur en espérance du réseau. Cela revient en pratique à calculer le gradient de $\mathbb{E}_b\left[\tilde{F}_\Theta(x)\right]$ par rapport à x.

L'espérance ne pouvant pas être calculée analytiquement, elle est estimée grâce à des méthodes Monte-Carlo. Pour évaluer la robustesse des réseaux randomisés, les auteurs de [1] proposent d'utiliser les attaques EoT plutôt que les attaques déterministes correspondantes. Dans le reste de ce papier, on utilise les attaques EoT pour évaluer la performance de notre approche.

4 Entraînement antagoniste randomisé

Bien que simple en apparence, l'entraînement antagoniste qui s'appuie sur des attaques PGD ou FGSM est un des seuls mécanismes de défense robuste en pratique [2]. Cette méthode a été étudiée en profondeur et montre des performances au niveau de l'état-de-l'art contre les attaques ℓ_∞. Malheureusement, cette méthode ne montre pas d'aussi bonnes performances contre les attaques ℓ_2 comme C&W. Dans cette partie, nous montrons dans un premier temps pourquoi l'entraînement antagoniste basé sur des attaques ℓ_∞, ne peut pas protéger les modèles contre des attaques ℓ_2 et vice-versa. Puis nous proposons RAT (pour *Randomized Adversarial Training*, ou entraînement antagoniste randomisé) qui est un mécanisme de défense qui combine l'entraînement antagoniste ℓ_∞ avec des techniques de bruitage pour offrir une protection simultanée contre les attaques ℓ_2 et ℓ_∞.

4.1 *No free lunch* pour l'entraînement antagoniste

Pour comprendre pourquoi l'entraînement antagoniste ℓ_∞ ne protège pas contre les attaques ℓ_2 nous avons d'abord besoin de caractériser les exemples malveillants générés par les attaques ℓ_∞.

Exemples malveillants ℓ_∞ : Soit ϵ la taille maximale de la perturbation générée en norme ℓ_∞, et r la perturbation d'un exemple $x \in \mathbb{R}^d$ telle que $\|r\|_\infty \leq \epsilon$. En calculant le signe du gradient de la fonction de perte, FGSM trouve des perturbations r dans le *coin* de la boule ℓ_∞ de rayon ϵ, comme illustré sur la Figure 1 (gauche). Ces perturbations restent visuellement imperceptibles parce que leur norme ℓ_∞ est bornée, mais leur taille est importante lorsqu'elles sont mesurées à l'aide de la norme ℓ_2 : $\|r\|_2 = \epsilon \times \sqrt{d}$. PGD^n, généralise FGSM en effectuant n appels successifs à FGSM. Les perturbations r générées, grâce à cette méthode, ont donc une norme $\|r\|_2 \geq \frac{\epsilon}{n}\sqrt{d}$.

Lorsque la dimension d est grande (ce qui est le cas pour la plupart des applications de l'apprentissage profond), toutes les perturbations sont regroupées autour des coins de la boule ℓ_∞, représentés en rouge dans la Figure 1, et très éloignés de la boule ℓ_2.

Étudions maintenant les exemples malveillants générés avec des attaques ℓ_2.

Exemples malveillants ℓ_2 : on choisit ϵ' comme étant la taille maximale de la perturbation mesurée en norme ℓ_2. En conséquence, tous les exemples malveillants générés par

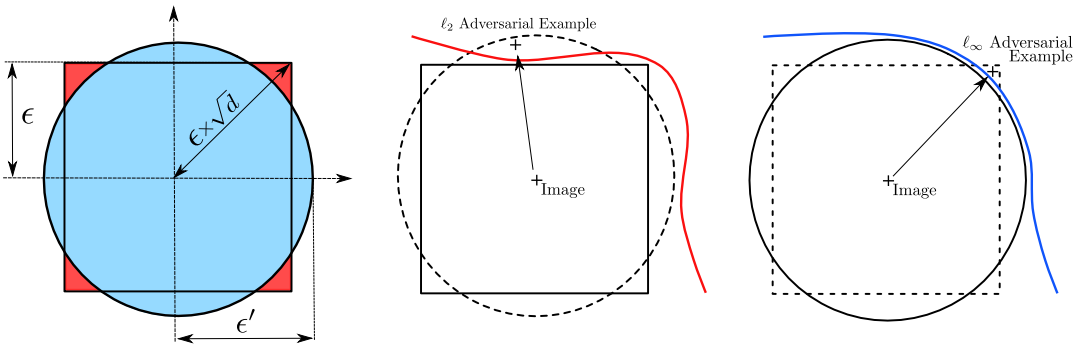


FIGURE 1 – Gauche : représentation en 2D des boules topologiques associées respectivement à la norme ℓ_∞ et ℓ_2 et pour des rayons respectifs ϵ et ϵ'. Milieu : un classifieur entraîné avec des perturbations ℓ_∞ (frontière de décision matérialisée par la ligne rouge) reste vulnérable aux attaques ℓ_2. Droite : un classifieur entraîné avec des perturbations ℓ_2 (frontière de décision matérialisée par la ligne bleue) reste vulnérable aux attaques ℓ_∞ (notons que ce cas n'est pas difficilement réalisable en pratique).

des attaques ℓ_2 doivent résider dans la boule ℓ_2 de rayon ϵ', représentée en bleu dans la Figure 1 (gauche). On remarque que la boule ℓ_2 n'inclut pas les coins de la boule ℓ_∞ à moins que l'on ne choisisse $\epsilon' \geq \epsilon\sqrt{d}$. En grande dimension (c'est à dire pour une grande valeur de d), la boule ℓ_2 n'inclura pas les coins de la boule ℓ_∞, à moins de choisir une très grande valeur pour ϵ', ce qui inclurait dans le même temps des perturbations largement visibles. En conséquence, l'ensemble des exemples ℓ_∞ et ℓ_2 sont mutuellement disjoints.

Comme les ensembles des exemples malveillants ℓ_∞ et ℓ_2 sont mutuellement disjoints, l'entraînement antagoniste avec des exemples ℓ_∞ ne permet de se protéger que contre les exemples ℓ_2 qui sont à l'intérieur de la boule ℓ_∞. Les autres exemples malveillants restent donc dangereux pour le modèle. Nous illustrons ce phénomène dans la figure 1 (milieu) : La croix au centre représente l'exemple original x et le carré autour de la croix représente l'ensemble des perturbations ℓ_∞ acceptables $\|r\|_\infty \leq \epsilon$. En entraînant un classifieur (matérialisé par la ligne rouge) à classifier x et toutes les perturbations ℓ_∞ correctement, on ne protège pas le classifieur contre les exemples malveillants qui se trouvent dans la *calotte* (i.e. les exemples qui se trouve dans la boule ℓ_2, mais en dehors de la boule ℓ_∞).

Remarquons qu'en grande dimension les coins de la boule ℓ_∞ sont très éloignés de la boule ℓ_2. En s'appuyant sur cette observation, nous faisons l'hypothèse que les exemples malveillants ℓ_∞ (qui se trouvent dans les coins de la boule ℓ_∞), ne couvrent véritablement qu'une toute petite fraction des exemples malveillants ℓ_2. Pour vérifier cette hypothèse, on mesure la proportion d'exemples malveillants dans la boule ℓ_2 après entraînement antagoniste ℓ_∞. Les résultats de cette expérience sont présentés dans la Figure 2 (gauche : sans entraînement antagoniste, droite : avec entraînement antagoniste).

Sur les deux graphiques, la région bleue représente la proportion d'exemples malveillants ℓ_2 qui sont à l'intérieur de la boule ℓ_∞. La région rouge, représente les exemples ℓ_2 malveillants qui se trouvent en dehors de la boule ℓ_∞, mais toujours à l'intérieur de la boule ℓ_2. Finalement, la région beige représente les exemples ℓ_2 malveillants qui se trouvent au delà de la boule ℓ_2. Le rayon ϵ' de la boule ℓ_2 varie de ϵ à $\epsilon \cdot \sqrt{d}$.

Sur le graphique de gauche (c'est-à-dire, sans l'entraînement antagoniste ℓ_∞), la plupart des exemples malveillants générés par C&W se trouvent dans la boule ℓ_∞. Sur le graphique de droite (c'est-à-dire, après l'entraînement antagoniste ℓ_∞), beaucoup moins d'exemples malveillants ℓ_2 se trouvent dans la boule ℓ_∞ (c'est l'effet escompté de l'entraînement antagoniste ℓ_∞). Mais la plupart des exemples malveillants ℓ_2 ont été déplacés dans la calotte, c'est-à-dire dans la zone qui se trouve à l'intérieur de la boule ℓ_2, mais en dehors de la boule ℓ_∞. Cependant la norme ℓ_2 de ces exemples, reste suffisamment faible, et l'entraînement antagoniste ℓ_∞ est donc sans conséquences positives pour la robustesse du réseau contre les attaques ℓ_2.

Remarquons également que même si l'entraînement antagoniste ℓ_2 est difficile à mettre en œuvre, il ne permettrait pas non plus de se protéger contre les attaques ℓ_∞ pour des raisons similaires (comme cela est illustré dans la Figure 1 (droite)).

Nous en concluons que les mécanismes de protection ℓ_2 et ℓ_∞ sont strictement complémentaires, et que les combiner est nécessaire pour pouvoir se protéger contre tout type d'attaques.

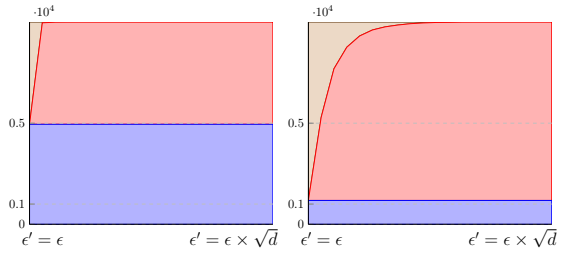


FIGURE 2 – Comparaison du nombre d'exemples malveillants produits par l'attaque C&W se trouvant dans la boule ℓ_∞ (zone bleu), se trouvant à l'extérieur de la boule ℓ_∞ mais dans le boule ℓ_2 (zone rouge) et se trouvant à l'extérieur des deux boules (zone grise). La valeur de ϵ est de 0.3 et ϵ' varie entre ϵ et $\epsilon \times \sqrt{d}$ sur l'axe des abscisses. A gauche : entraînement classique, a droite : entraînement antagoniste. Entre les deux situation, la plupart des exemples malveillants se sont déplacés de l'intérieur de la boule ℓ_∞ vers l'extérieur tout en restant dans la boule ℓ_2.

4.2 RAT & LeRAT

Pour mettre en place une stratégie de défense qui permet de se défendre à la fois contre les attaques ℓ_2 et ℓ_∞, nous proposons une nouvelle méthode, nommée RAT (Randomized Adversarial Training) et décrite par l'Algorithme 1. Le problème d'optimisation est le suivant :

$$\min_\theta \mathbb{E}_{(x,y)} \left[\max_{x'/||x'-x||\leq\epsilon} \mathbb{E}_b \left[\mathcal{L}(\tilde{F}_\theta(x'), y) \right] \right]$$

b étant le bruit injecté dans le réseau $\tilde{F}_\theta$.

RAT diffère fondamentalement de l'apprentissage antagoniste car il a été conçu pour entraîner des algorithmes randomisés. Rappelons [1] qu'à chaque fois que l'on fait appel à un algorithme randomisé (e.g. pour calculer une prédiction ou un gradient) un nouveau vecteur de bruit est généré et ajouté sur l'activation de la première couche du réseau. Pendant l'apprentissage de RAT, le bruit est généré à la fois pendant le calcul de l'attaque (ligne 5 dans l'Algorithme 1) et la mise à jour de paramètres par rétro-propagation (ligne 7). Notons que par souci d'efficacité, nous n'utilisons pas la méthode EoT pendant la phase d'apprentissage.

LeRAT (Learned Randomized Adversarial Training) : *LeRAT* est une variation naturelle de la méthode RAT dans laquelle les paramètres du bruit font partie des paramètres mis à jour pendant l'apprentissage. Cette approche est motivée par le fait que l'apprentissage des paramètres du bruit généré pendant l'apprentissage du réseau devrait permettre de trouver les paramètres qui représentent un bon compromis entre précision et défense. Notons que cette procédure ne peut être mise en place qu'en utilisant une fonction de coût antagoniste car apprendre les paramètres avec une fonction de coût classique nous mènerait simplement à annuler le bruit.

1. voir définition 2

Algorithm 1 RAT

1: Initialiser θ
2: **for** $i \in [1, nb\_steps]$ **do**
3: $B_i := (x_k, y_k)_k$ le mini-batch
4: **for** $x_k \in B_i$ **do**
5: Calculé l'attaque malveillante x'_k de l'image x_k contre $\tilde{F}_\theta$ avec une méthode arbitraire
6: **end for**
7: Mettre à jour les paramètres θ avec le critères suivantes : $\frac{1}{|B_i|} \sum_k \mathcal{L}(\tilde{F}_\theta(x'_k), y_k)$
8: **end for**
9: **return** Le réseau $\tilde{F}_\theta$

5 RAT, le robuste

Dans cette section, nous analysons d'abord la robustesse des réseaux de neurones entraînés avec RAT contre les attaques ℓ_∞. Puis, nous discutons des performances contre des attaques ℓ_2 plus puissantes et donnons une interprétation empirique pour mieux comprendre les mécanismes de défense contre ce type d'attaques.

Pour mesurer la robustesse des mécanismes de protection contre les attaques ℓ_∞ et ℓ_2, nous suivons le protocole expérimental proposé par [2], nous évaluons la précision des réseaux randomisés face à des attaques EoT. Nous considérons des attaques non ciblées (i.e. une mauvaise classification d'une classe à n'importe quelle autre est considérée comme un échec pour le défenseur) dans un cadre "white-box" (i.e. l'attaquant a accès à toutes les informations du réseau). Il s'agit du protocole le plus conservateur pour évaluer les mécanismes de défense. Tous les mécanismes de défense ont été entraînés par nos soins. Voir Section 5.4 pour une description plus détaillée de notre protocole expérimental.

5.1 Robustesse contre les attaques ℓ_∞

Dans cette partie, nous comparons l'efficacité des mécanismes de protection RAT et LeRAT avec l'entraînement antagoniste ℓ_∞ et les réseaux randomisés proposés par [15]. Les mécanismes de défense sont testés contre des attaques ℓ_∞ générées par FGSM et PGD [13]. Les résultats de cette expérience sont présentés dans la Figure 3.

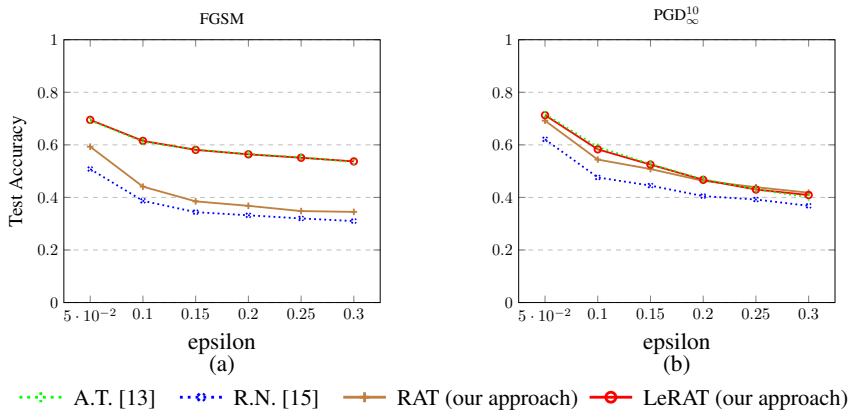


FIGURE 3 – Ces figures représentent la précision sur notre partie de validation du jeu de donnés CIFAR10 en fonction de la valeur maximale autorisée (ϵ) pour la norme ℓ_∞ de la perturbation. Ces valeurs variés de $5 \cdot 10^{-2}$ à 0.3 et plus la valeur est grande, plus l'image est visuellement éloignée de l'image originale. On remarque que l'entraînement antagoniste obtient des performances équivalentes à LeRAT et que RAT surpasse l'entraînement bruité en termes de précision.

De manière peu surprenante, la précision de tous les mécanismes de défense diminue quand la valeur de ϵ augmente de 0.005 à 0.03 (i.e. les exemples sont plus perturbés). Dans ce cadre, LeRAT obtient les meilleurs résultats de robustesse en atteignant l'état de l'art dans le cas ℓ_∞. R.N. a des performances moins bonnes dans ce cas mais RAT obtient des performances correctes en particulier contre les attaques PGD. Dans un second temps, nous fixons le paramètre ϵ à 0.1 et augmentons le nombre des simulations Monte-Carlo utilisées pour les attaques EoT (plus de simulations rendent les attaques plus puissantes). Les résultats sont illustrés par la Figure 4 (a) et (b). Comme on peut le constater, R.N. offre un bas niveau de protection contre les attaques EoT puissantes. En comparaison, LeRAT n'est pas impacté par les attaques plus puissantes et RAT offre un bon compromis.

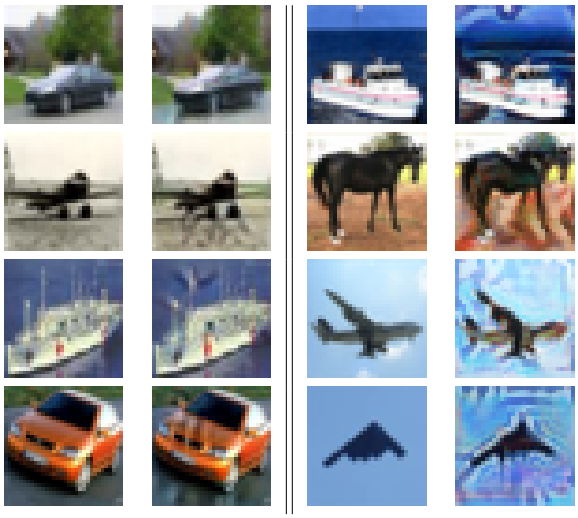


TABLE 1 – Images avant et après perturbation. Les perturbations sont calculées par l'attaque C&W sur un réseau défendu par RAT. Sur la gauche (la norme ℓ_2 de la perturbation est égale à 4), on peut différencier facilement les deux images. Sur la droite, (la norme ℓ_2 de la perturbation est égale à 10), on peut détecter que l'image est perturbée sans comparer avec l'originale.

5.2 Robustesse contre les attaques ℓ_2

Nous évaluons maintenant nos méthodes contre les attaques ℓ_2 C&W. Étant donné une image d'entrée, l'attaque C&W trouve la perturbation minimale qui va mener le réseau à mal classifier. Nous évaluons notre approche en utilisant le même cadre que dans [2], et discutons d'autres aspects intéressants de cette évaluation. Les résultats sont présentés dans la Figure 4 (c).

Tout d'abord, nous remarquons que l'entraînement antagoniste (A.T.) n'offre aucune protection contre les attaques ℓ_2. Nous remarquons également que les réseaux randomisés sont bien plus performants que A.T. entraîné avec des exemples d'attaques ℓ_∞. Comme expliqué dans la Section 4, l'entraînement antagoniste ne peut pas offrir une bonne protection contre les attaques ℓ_2. Enfin, la robustesse de LeRAT est significativement plus basse que les deux autres approches. Remarquons que dans cette expérience, la norme de la perturbation n'est pas bornée, par conséquent il n'y a aucune garantie que l'attaque malveillante soit en dessous du seuil

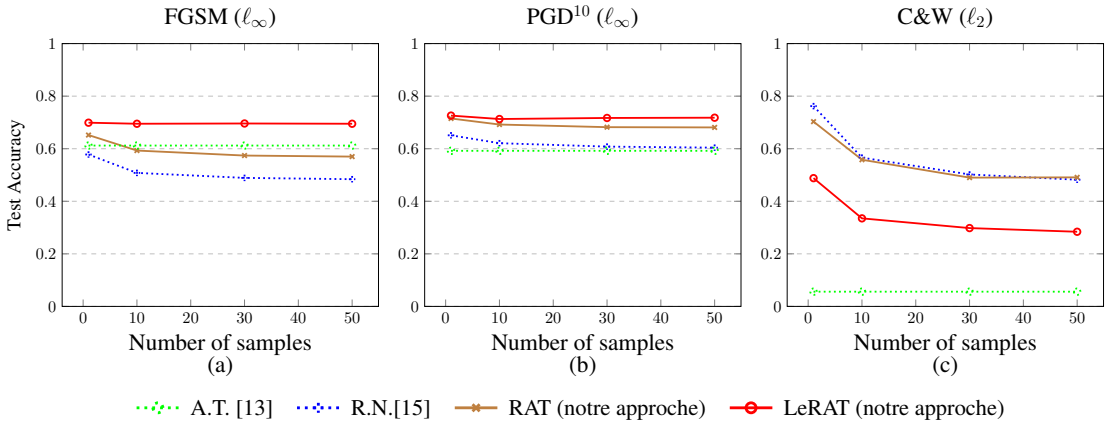


FIGURE 4 – Ce graphique représente la précision de différents modèles de défense respectivement attaqués par les attaques FGSM ℓ_∞ (a), PGD^{10} ℓ_∞ (b) and C&W ℓ_2 (c). Toutes les attaques sont réalisées en utilisant la méthode EoT, estimée avec des méthodes de Monte-Carlo à taille d'échantillonnage croissant. Les performances de LeRAT dépassent les autres approches sur les attaques de norme ℓ_∞. RAT et l'entraînement bruité sont plus efficaces contre les attaques de norme ℓ_2. On remarque que RAT offre le meilleur compromis de défense.

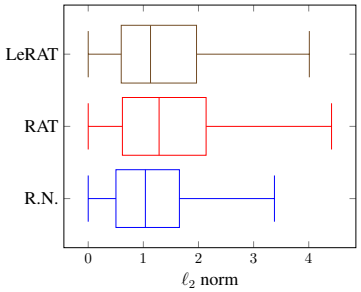


FIGURE 5 – Ce graphique (boîtes à moustache) compare les normes ℓ_2 des exemples malveillants produits par l'attaque C&W sur des réseaux défendus par LeRat, RAT et entraînement bruité. Les exemples malveillants générés par C&W pour attaquer le réseau défendu par RAT ont une norme plus importante (et sont ainsi plus facilement identifiable) que ceux générés pour les autres défenses.

de perception humain. Dans la Figure 5, nous avons tracé différentes statistiques à propos des attaques malveillantes générées par C&W pour tromper chaque mécanisme de défense.

De façon surprenante, les statistiques de la Figure 5 montrent que la moyenne des normes ℓ_2 nécessaires pour tromper le réseau sont élevées. Dans la Figure 1, nous montrons différents exemples malveillants de norme ℓ_2 variable. Comme nous le constatons, les exemples malveillants avec une norme ℓ_2 supérieure à 4 trompent le mécanisme de défense mais sont facilement identifiables à l'oeil humain. En d'autres termes, tromper ces mécanismes de défense peut nécessiter une perturbation beaucoup trop grande pour rester indiscernable à l'oeil humain.

5.3 Performance contre des attaques fortes et discussion

Dans cette section, nous nous intéressons à la performance des différents mécanismes contre des attaques plus puissantes. Pour ℓ_∞, nous utilisons l'attaque PGD^{10} avec une valeur maximale pour ϵ égale à 0.3. Dans le cas ℓ_2, nous utilisons C&W avec 40 itérations. Dans les deux cas, nous utilisons une estimation de Monte-Carlo avec 50 tirages pour obtenir une attaque EoT. Les résultats sont synthétisés dans la Table 2. Nous remarquons que RAT et LeRAT offrent les meilleures performances en terme de robustesse contre les attaques ℓ_∞ et que RAT est le plus performant dans le cas ℓ_2. RAT bat de manière nette les performances des meilleurs mécanismes actuels et offre le meilleur compromis.

5.4 Details sur la méthodologie d'évaluation

Jeu de données Pour nos expériences, nous utilisons le jeu de données CIFAR10. Il est constitué de 60 000 32x32 images couleur avec 10 classes. Le jeu de données est séparé en 50 000 images pour l'entraînement et 10 000 images pour l'évaluation.

Modèle de base Nous avons utilisé une architecture Wide ResNet [22] pour toutes nos expériences avec 10 en pa-

Model	ℓ_∞ **Attack**	ℓ_2 **Attack**
Adv Training (A.T.) [13]	0.400	0.056
Randomized Network (R.N.) [15]	0.356	0.489
Randomized Adv Training (RAT)	**0.409**	**0.499**
Learned Randomized Adv Training (LeRAT)	0.400	0.296

TABLE 2 – Comparaison de la robustesse de toutes les approches contre des attaques ℓ_∞ et ℓ_2. L'entraînement antagoniste offre une bonne protection contre les attaques ℓ_∞ mais fonctionne médiocrement contre des attaques ℓ_2. L'attaque ℓ_∞ est obtenue via PGD^{10} avec $\epsilon = 0.3$ et l'attaque ℓ_2 via 40 itérations de C&W. Pour toutes les attaques contre des réseaux randomisés, on effectue une estimation de Monte-Carlo avec 50 tirages pour obtenir une estimation en espérance de la valeur du réseau. Les réseaux randomisés offrent une bonne protection contre les attaques ℓ_2 mais restent sous-optimales contre les attaques ℓ_∞. Notre approche (RAT) offre la meilleure protection contre toutes les attaques.

ramètre de largeur et 28 en paramètre de profondeur. Toutes les valeurs des pixels ont été normalisées entre -1 et 1. Notre modèle est appris sur 200 époques par paquets de 200 images avec un pas de gradient initial de 0.1. Durant l'entraînement, le pas de gradient diminue à 0.02, 0.004 et 0.00008 respectivement après 7500, 15000 and 20 000 itérations. Nous utilisons aussi des "random crop" et "random flip" pour augmenter le jeu de données. Comme régularisation nous utilisons du "dropout" et un "weight decay" de 0.002. Pour entraîner le réseau, nous utilisons l'entropie croisée comme fonction de coût et un momentum de 0.9 pour la descente de gradient stochastique. Ce réseau obtient 95% de précision sur CIFAR10.

Entraînement antagoniste avec bruit. Pour l'entraînement antagoniste, nous utilisons la procédure décrite dans l'Algorithme 1 avec *Projected Gradient Descent (PGD)* comme attaque. PGD est implémenté avec 1 itération, une perturbation globale de 0.03 et un pas de gradient de 0.02 Pour choisir la variance du bruit à injecter, nous avons conduit une optimisation d'hyperparamètres par *grid search*. Une loi Gaussienne de moyenne 0 et de variance 1 offre un bon rapport précision/robustesse.

Evaluation sous attaques. Pour évaluer nos attaques, nous calculons la précision sous attaques (précision face aux exemples malveillants) avec FGSM [7], PGD [13], comme attaques ℓ_∞et C&W [3] comme ℓ_2. Pour PGD, nous utilisons 10 itérations avec un pas de gradient de 0.02. Pour C&W, nous utilisons 9 itérations pour la recherche binaire d'hyperparamètres et 40 iterations au total. Comme décrit dans [1], les attaques contre les défenses randomisées doivent être calculées avec la valeur moyenne de la fonction de coût ou des logits. Dans nos expériences, nous estimons ces moyennes par simulation de Monte Carlo.

6 Conclusion

Nous avons mené une analyse expérimentale et qualitative pour mieux comprendre le comportement des différents mécanismes de défense soumis à différentes attaques. A partir de ces observations, nous avons construit deux nouvelles techniques appelées RAT et LeRAT pour entraîner des réseaux robustes contre des attaques ℓ_2 et ℓ_∞ que nous avons comparées à l'état de l'art : l'entraînement antagoniste contre les attaques ℓ_∞ et les réseaux randomisés contre les attaques ℓ_2. Nos expériences montrent que notre approche surclasse les performances des méthodes de l'état de l'art, constituant ainsi une défense plus générale et plus robuste que les approches existantes.

Références

[1] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. *arXiv preprint arXiv :1707.07397*, 2017.

[2] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security : Circumventing defenses to adversarial examples. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 274–283, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[3] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[4] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, and A. Madry.

On evaluating adversarial robustness. *arXiv preprint arXiv :1902.06705*, 2019.

[5] G. S. Dhillon, K. Azizzadenesheli, J. D. Bernstein, J. Kossaifi, A. Khanna, Z. C. Lipton, and A. Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018.

[6] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard. Robustness of classifiers : from adversarial to random noise. In *Advances in Neural Information Processing Systems*, pages 1632–1640, 2016.

[7] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[9] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[10] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv :1607.02533*, 2016.

[11] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

[12] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh. Towards robust neural networks via random self-ensemble. In *European Conference on Computer Vision*, pages 381–397. Springer, 2018.

[13] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[14] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool : a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[15] R. Pinot, L. Meunier, A. Araujo, H. Kashima, F. Yger, C. Gouy-Pailler, and J. Atif. Theoretical evidence for adversarial robustness through randomization : the case of the exponential family. *arXiv preprint arXiv :1902.01148*, 2019.

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAi, 2018.

[17] A. S. Rakin, Z. He, and D. Fan. Parametric noise injection : Trainable randomness to improve deep neural network robustness against adversarial attack. *arXiv preprint arXiv :1811.09310*, 2018.

[18] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-GAN : Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

[19] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman. Pixeldefend : Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018.

[20] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[21] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018.

[22] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv :1605.07146*, 2016.

Défense contre les exemples adversaires : L_1 Double rétropropagation

Ismaila Seck[1,2] , Gaëlle Loosli[2,3] , et Stéphane Canu[1]

[1]Normandie Univ, INSA Rouen, UNIROUEN, UNIHAVRE, LITIS, France
[2]UCA - LIMOS UMR 6158 CNRS
[3]PobRun, Brioude, France

Résumé

Adversarial examples are a challenging open problem for deep neural networks. We propose in this paper to add a penalization term that forces the decision function to be flat in some regions of the input space, such that it becomes, at least locally, less sensitive to attacks. Our proposition is theoretically motivated and shows on a first set of carefully conducted experiments that it behaves as expected when used alone, and seems promising when coupled with adversarial training.

Mots-clef : Adversarial training, gradient penalisation.

1 Introduction

Deep learning algorithms have set the state-of-the-art in several domains among which image classification. However, [GSS15] showed that it is possible, and relatively easy, to fool Deep Neural Networks by adding to the inputs a particular perturbation. This added perturbation may be such that the disturbed inputs and the original ones are very close according to some metrics, but are assigned to different classes. Several defense mechanisms have been introduced to prevent such a behavior, but their efficiency is limited to specific cases so that the general problem of preventing the existence of adversarial examples remains open. One idea is to have a constant output over a region around known training points. For a differentiable function, that means an arbitrarily chosen norm of the output's gradient with respect to the input should be 0 or at least as small as possible over that region. Our claim is that by using adversarial training and by penalizing the gradient's norm of the output with respect to the input, the robustness of the model can be improved. The L_1-norm is chosen and this choice will be theoretically motivated by calculus.

2 Related work

The adversarial examples were first presented in [SZS+13], and the principle of adversarial training was introduced at the same time. Adversarial training consists in augmenting the dataset with potentially adversarial points. But it was impractical, since the method used to generate adversarial samples, L-BFGS, was too slow. Adversarial training became more convenient to use with the introduction of Fast Gradient Sign Method (FGSM) [GSS15], which is much faster and generates adversarial examples with a good success rate. Moreover, using a first order expansion of Taylor series, adversarial training can be seen as an ℓ_1-norm penalty of the derivative of the loss with respect to the inputs [SGOS+18]. Note that *regularization functional which penalize derivatives of the resulting classifier function are not typically used in deep learning* [HA17]. The idea of penalizing the gradient of the output with respect to the input was introduced by [DLC92] under the name *double backpropagation* and later used in [HS95] to find large connected regions of the error function called flat minima. The ultimate goal was the improvement of the generalization of their models, which differs slightly from our goal here. In *double backpropagation*, the ℓ_2-norm of the loss is penalized. Using the Energy loss function, it was stressed that the penalization of the loss would have little to no effect when the classification is good. To balance out that effect, the multiplicative parameter of the penalization ought to be large enough.

The difference between double backpropagation and our gradient penalization is that we penalize the ℓ_1-

norm of the gradient of each output with respect to the input while, in backpropagation, the ℓ_2-norm of the loss is penalized. Hence, we should not have the problem that occurs when the penalization term is multiplied by a small error vector. Nevertheless, this might not be a problem when using a different loss function. Although empirical evidence show double backpropagation's efficiency to enhance generalization, it is insufficient to defend against adversarial examples. That is what [PMW+16] highlights saying *limiting sensitivity to infinitesimal perturbation [e.g., using double backpropagation DLC92] only provides constraints very near training examples, so it does not solve the adversarial perturbation problem.* But there are evidence that coupling that gradient penalty with adversarial training increases the robustness.

3 Defensive gradient penalty

It has been shown that that maliciously crafted examples can fool the deep learning classifiers and lead them to misclassify those examples with high confidence. In addition to the adversarial training, a gradient penalization is proposed here to make the models more robust. Let $\mathbf{x}$ be an input image stored in a vector belonging to the input space $\mathcal{X} = [0,1]^d$, where d is the dimension of $\mathbf{x}$ and let y be the target, a one-hot label, associated with $\mathbf{x}$. Let f, a differentiable function, $f : \mathcal{X} \to \mathbb{R}^c$ where c is the number of classes, such that the decision function $D(x) = argmax(f(x))$ represents our classifier. And let $\mathcal{L}(f(\mathbf{x}), \mathbf{y})$ be a common differentiable loss function.

For a given input $\mathbf{x}$, a perturbation direction $\mathbf{v}$ and a positive scalar ε, the first order expansion of the transfer function i-th component, f_i, of the neural network is :

$$f_i(\mathbf{x} + \varepsilon\mathbf{v}) = f_i(\mathbf{x}) + \varepsilon\mathbf{v}^{\mathrm{T}}\nabla_{\mathbf{x}} f_i(\mathbf{x}) + \circ(\varepsilon\|v\|). \quad (1)$$

Considering the FGSM attack [GSS15] with $\mathbf{v} = \mathrm{sign}(\nabla_{\mathbf{x}}\mathcal{L}(f(\mathbf{x}), \mathbf{y})$, we have $\mathbf{v}^{\mathrm{T}} = \{\pm 1\}^d$, $|f_i(\mathbf{x}+\varepsilon\mathbf{v}) - f_i(\mathbf{x})| \approx |\varepsilon\mathbf{v}^{\mathrm{T}}\nabla_{\mathbf{x}} f_i(\mathbf{x})| \leq \varepsilon\|\nabla_{\mathbf{x}} f_i(\mathbf{x})\|_1 = \varepsilon w^{\mathrm{T}}\nabla_{\mathbf{x}} f_i(\mathbf{x})$ for $w = \mathrm{sign}(\nabla_{\mathbf{x}} f(\mathbf{x}))$. So that, in this case, finding weights that minimize sensitivity to infinitesimal perturbation of the input can be done by minimizing the ℓ_1-norm of the gradient of each component of the transfer function with respect to the input. Our approach is therefore to penalize the ℓ_1-norm of the gradient of each coordinate $f_i(x)$ not only on the original point but also on the potentially adversarial point generated. If we manage to have 0 as the ℓ_1-norm of those gradients at two points that are very close, then the output of the classifier is almost constant along the segment joining those two points. Indeed we have :

$$0 \leq |f_i(\mathbf{x}+\varepsilon\mathbf{v}) - f_i(\mathbf{x})| \leq \varepsilon \sup_{t\in[0,\varepsilon]} \|\nabla_{\mathbf{x}} f_i(\mathbf{x}+t\mathbf{v})\|_1 \quad (2)$$

The regularization coefficient being small (ε has to be small for $\mathbf{x} + \varepsilon\mathbf{v}$ to be considered as an adversarial example) the regularization is not used at its full potential. In order to increase the regularization coefficient, an explicit penalization of the ℓ_1-norm is added to the loss function. Hence the analogy with [DLC92], which penalize the ℓ_2-norm of the gradient of the loss while in this paper the sum of the ℓ_1-norm of the outputs $(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_c(\mathbf{x}))$ with respect to the input is directly penalized. But, it is very hard to make derivatives very small using only the penalization of the gradient when penalizing on the training points only, and it is also ineffective against adversarial examples.

Better results are obtained when we penalize the gradients on the training set and also on the adversarial example near the training set. Doing so each training point is associated with an adversarial example, as in classical adversarial training, and we penalize the gradient on both these points. Let $\mathbf{x}$ be a point of the training set and $\mathbf{x}_{adv}$ the adversarial example computed from $\mathbf{x}$ using the FGSM. Let us assume that we train the classifier for long enough so that $\|\nabla_{\mathbf{x}} f_i(\mathbf{x})\|_1 \approx 0$ and $\|\nabla_{\mathbf{x}} f_i(\mathbf{x}_{adv})\|_1 \approx 0$ for all $i = 1, \ldots, c$, since those two points are close enough, the variation of f on the segment joining $\mathbf{x}$ and $\mathbf{y}$ is so small that the class does not change. The ideal training progress is presented in figure 1. See section 4.2 for comparison of the results of between classical adversarial training, and the new variant introduced in this paper. The loss function used is :

$$\begin{aligned}\mathcal{L}_{gp}(\mathbf{x}, \mathbf{y}) &= \mathcal{L}(f(\mathbf{x}), \mathbf{y}) + \lambda \sum_{i=1}^{c} \|\nabla_{\mathbf{x}} f_i(\mathbf{x})\|_1 \\ &= \mathcal{L}(f(\mathbf{x}), \mathbf{y}) + \lambda \sum_{i=1}^{c}\sum_{j=1}^{d} |J_{ij}| \\ &= \mathcal{L}(f(\mathbf{x}), \mathbf{y}) + \lambda \|J\|_{1,1}, \end{aligned} \quad (3)$$

where d denotes the dimension of the input image, c, the number of neurons on the output layers of the neural network *i.e.* the number of classes, λ, the penalization parameter, $J_{ij} = \frac{\partial f_i(\mathbf{x})}{\partial x_j}$, the Jacobian matrix of f and $\|.\|_{1,1}$, the entry wise L_1-norm.

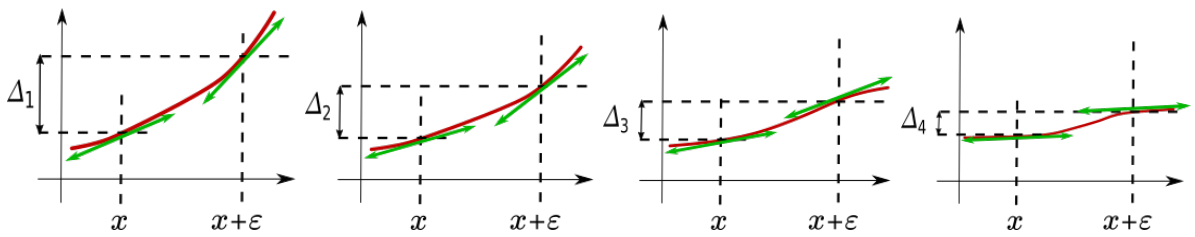


FIGURE 1 – Figure showing the ideal progression of the training for one training point x, and associated adversarial points. At first, the difference $\Delta = f(x+\varepsilon) - f(x)$ of ordinates and gradients norm at x and $x+\varepsilon$ are high. During the course of the training, Δ decreases, making the value at x and $x+\varepsilon$ closer. The gradients' norm also decrease. In the end, we have almost the same value for f at those points, and the gradients are nearly 0. In those conditions, we have $\Delta_1 > \Delta_2 > \Delta_3 > \Delta_4 \approx 0$, and the variation between x and $x+\varepsilon$ is then very small.

model A	model B	model C
conv(64,8,2)	conv(128,3,1)	FC(512)
conv(128,6,2)	conv(64,3,2)	FC(256)
conv(128,5,1)	FC(128)	FC(128)
FC(10)	FC(10)	FC(10)
710,218	1,460,938	567,434

TABLE 1 – Description of the models used in this paper : conv(nf, k, s) represents a convolutional layer with nf filters, of size $k \times k$ applied with a stride s. FC(nn) represents a fully connected layer with nn neurons. All activations are ReLU except the output layer. The numbers in the last line represent the number of parameters.

4 Experimental Results

All experiments here are built upon 3 models (referred as A,B and C for simplicity) that are described in table 1. The first experiments is our proof of concept. It shows that the penalization helps the defense. Since alone it's still not enough to propose a robust model, the second experiment explores the coupling with adversarial training.

4.1 Proof of concept

The goal of this experiment is to show that penalizing the gradient improves a lot the robustness of the model while keeping the efficiency on the clean data, and the more we penalize, the more the effect is visible. In our tests, we observe that we double (at least) the number of correctly classified adversarial examples.

We attack the model using the FGSM in a white-box setup in which the attacker has access to the parameters of the model and to the test set. The value $\varepsilon = 0.3$ is chosen as the intensity of the pixel wise perturbation allowed to the attacker as it is often the case for the data used in this experiment. Several values of λ are used, allowing to have a notion of the influence of that hyper-parameter on the performance on adversarial samples. The MNIST dataset and three different architectures are used. The training and testing split of Keras is used, and the training set is then split into two groups, one of 55000 used to train models and the remaining 5000 are used as a validation set. Models are trained until their accuracy on the validation set stop increasing (difference of accuracy between 2 epochs less than 0.0001) after 10 epochs, or until 100 epochs of training on the clean data is achieved. This process is repeated 10 times, the mean value and the standard deviation are recorded in table 2. The program always terminated before 100 hundred epochs were reached, around the 20th epoch. The optimization method used is Adam with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, a label smoothing of 0.1 was also used.

4.2 Coupling with adversarial training

The previous experiments show that our penalization has an interesting effect on the robustness of the models. Now we explore the coupling with adversarial training, and the hope is that the penalization has the same impact, in order to obtain more robust models than adversarial training alone. We train the same model using adversarial training with the classical loss function and with the gradient penalty in addition to that loss. We train for one hundred epochs. The adversarial training consists in adding adversarial examples to the original training set to make the models more robust. Typically, the FGSM is used to generate those adversarial examples due to its practicality. So it will

0. The loss function used here is not exactly the one proposed in 3. Instead of $\|J\|_{1,1}$, $\sum_{j=1}^{d} |\sum_{i=1}^{c} \frac{\partial f_i(\mathbf{x})}{\partial x_j}|$ is used due to time constraints.

λ		50	25	10	1	0.1
A	clean	99.37 ± 0.04	99.36 ± 0.03	99.38 ± 0.04	99.35 ± 0.03	99.35 ± 0.06
	adv	45.09 ± 5.36	44.96 ± 4.44	46 ± 10.16	28.83 ± 9.34	22.64 ± 5.02
B	clean	98.99 ± 0.05	98.99 ± 0.05	98.98 ± 0.07	98.94 ± 0.07	98.90 ± 0.05
	adv	6.35 ± 3.258	4.69 ± 1.91	3.69 ± 0.73	3.00 ± 1.14	3.17 ± 0.82
C	clean	98.63 ± 0.11	98.67 ± 0.05	98.57 ± 0.06	98.55 ± 0.05	98.60 ± 0.08
	adv	29.92 ± 14.24	25.76 ± 11.96	18.24 ± 9.97	19.99 ± 5.63	13.66 ± 6.64

TABLE 2 – Accuracy on clean test and adversarial test point of model. We observe that the more we penalize, the more robust the model becomes. However, while the gain is significant, it's clearly not enough to call it a robust model.

ε		0.05		0.1		0.2		0.3	
		clean	adv	clean	adv	clean	adv	clean	adv
A	adv_train	99.36 ±0.05	98.57 ±0.11	99.34 ±0.06	97.50 ±0.26	99.27 ±0.03	96.39 ±0.37	99.30 ±0.06	95.72 ±0.19
	adv_train_gp	99.38 ±0.028	**98.73** ±0.072	99.35 ±0.035	97.45 ±0.30	99.25 ±0.061	96.63 ±0.30	99.26 ±0.06	**97.60** ±0.34
B	adv_train	98.94 ±0.079	98.37 ±0.48	99.07 ±0.03	98.22 ±0.61	98.96 ±0.05	98.55 ±0.26	98.85 ±0.05	98.67 ±0.07
	adv_train_gp	99.05 ±0.05	98.18 ±0.15	99.06 ±0.05	97,64 ±1.04	98.95 ±0.02	98.30 ±0.49	98.84 ±0.07	97.80 ±1.11
C	adv_train	98.98 ±0.06	96.26 ±0.22	98.68 ±0.06	93.56 ±0.47	98.59 ±0.07	93.24 ±0.48	98.48 ±0.1	95.52 ±0.74
	adv_train_gp	98.79 ±0.04	96.36 ±0.11	98.87 ±0.07	**94.56** ±0.16	98.69 ±0.09	93.73 ±0.75	98.42 ±0.08	**96.62** ±0.51

TABLE 3 – Table showing the performance of models A, B and C, trained with adversarial training and adversarial training plus gradient penalty ($\lambda = 10$)We can see that, the adv_gp performs better for model A and $\varepsilon = 0.3$, and is sensibly equal for other values of ε. The large standard deviation values might be an indication that the training was still in progress when 100th epochs was reached.

be used in the following experiments with a clipping to make sure that adversarial examples remains in $\mathcal{X}$. During training, for each image of a batch, an adversarial image is generated and added to the batch with the same target as the original image associated. A label smoothing of 0.1 is also used for this experiment.

Table 3 shows results for $\lambda = 10$. It turns out that performances are not as impressive as they were without adversarial training, even though for some settings, the penalization helps. One reason could be that λ is not high enough. However for computing time reason, we could not conduct the same set of experiments with different values at the same level of care. Hence we propose some partial results to argue our point in table where we see that increasing λ provides better results. We note also that we increased the number of epoch up to 150.

λ	0	10	50	100	200	1000
clean	98.48 ±0.10	98.42 ±0.08	98.49 ±0.06	98.59 ±0.10	98.60 ±0.11	98.62 ±0.07
adv	95.52 ±0.74	96.62 ±0.51	**97.25** ±0.05	96.88 ±0.51	96.71 ±0.19	94.65 ±0.79

TABLE 4 – For model C, we fixed $\varepsilon = 0.3$ and different λ, on the same setting as previous experiment. $\lambda = 0$ refers to the adversarial learning alone. We observe improvement when λ increases, until it fails when too high.

5 Conclusion and future work

In this paper we propose to improve deep neural networks' robustness to adversarial examples by adding a penalization term. We show that penalizing the gradients of the output with respect to input, or in other words the Jacobian, can have an effect defending against adversarial but does not suffice. Coupled with adversarial training to give L_1-norm double backpropopagation adversarial defense, we provide good hints that it is still a good approach. In future work we will extend the experimental part to more sets of parameters and other datasets.

Références

[DLC92] Harris Drucker and Yann Le Cun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6) :991–997, 1992.

[GSS15] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[HA17] Matthias Hein and Maksym Andriush-

chenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pages 2266–2276, 2017.

[HS95] Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. In *Advances in neural information processing systems*, pages 529–536, 1995.

[PMW+16] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[SGOS+18] Carl-Johann Simon-Gabriel, Yann Ollivier, Bernhard Schölkopf, Léon Bottou, and David Lopez-Paz. Adversarial vulnerability of neural networks increases with input dimension. *arXiv preprint arXiv :1802.01421*, 2018.

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv :1312.6199*, 2013.

Vers un désenchevêtrement de l'ambiguïté de la tâche et de l'incertitude du modèle pour la classification avec option de rejet à l'aide de réseaux neuronaux*

Titouan Lorieul[†1] et Alexis Joly[2]

[1]Zenith, LIRMM, Université de Montpellier, Inria, France
[2]Zenith, Inria, Sophia-Antipolis, France

Résumé

La classification avec option de rejet est un moyen d'aborder le problème de l'estimation de l'incertitude d'un classifieur. Les approches récentes s'attaquant à ce problème utilisent des critères basés, soit, sur une mesure de confiance, soit, sur une mesure de dispersion. Cependant, aucune d'entre elles ne combine explicitement les deux principales sources d'incertitude : l'ambiguïté de la tâche, intrinsèque à celle-ci, et l'incertitude du modèle, découlant de l'échantillonnage des données et de la stochasticité de l'apprentissage. Dans cet article, nous explorons comment ces deux quantités pouvent être fusionnées afin d'établir des critères de rejet plus efficaces. En particulier, nous proposons une série de méthodes combinant des mesures de désaccord et des estimations de l'ambiguïté en utilisant un ensemble de modèles. Des expériences sur des jeux de données de classification synthétiques construits pour modéliser différents types d'incertitudes indiquent que ces nouveaux critères ont des performances similaires aux méthodes de référence. Néanmoins, des analyses plus approfondies montrent des indices empiriques qui mettent en avant l'existence d'information supplémentaire dans la distribution des résultats de l'ensemble. Dans les faits, le réjecteur idéal peut être une fonction plus complexe que les critères précédents, et peut même parfois être contre-intuitif, ce qui implique qu'il est difficile de trouver un critère théorique de rejet qui généraliserait à travers les tâches et que cela reste un problème ouvert.

Mots-clef : Classification avec option de rejet, estimation d'incertitude, réseaux neuronaux, ensemble de modèles.

*Travaux soumis à la conférence ECML-PKDD 2019.

†titouan.lorieul@inria.fr

1 Introduction

Il est important, sinon nécessaire, de disposer de mesures précises de l'incertitude des prévisions d'un modèle dans de nombreux scénarios pratiques où l'on ne peut pas se permettre de commettre des erreurs. Cela est en particulier vrai dans des applications médicales, de conduite autonome, etc. Cependant, quantifier précisément cette information d'incertitude est un problème difficile, surtout lorsque le processus d'apprentissage n'est pas entièrement compris comme cela est le cas pour les réseaux neuronaux. Une façon d'assouplir cet objectif ambitieux, tout en progressant dans cette direction, est de permettre aux classifieurs de refuser de donner une réponse pour une entrée donnée. Ceci est connu sous le nom de *classification avec une option de rejet* [Cho70, HW06]. Cette décision de rejeter peut tirer parti de l'incertitude prédictive sans avoir à la modéliser complètement et ainsi permettre de mieux comprendre ce qui est nécessaire pour construire des modèles fournissant des informations d'incertitude plus complètes.

La classification avec option de rejet consiste à fournir deux fonctions : un prédicteur $h : \mathcal{X} \to \mathcal{Y}$ et un réjecteur $r : \mathcal{X} \to \{0, 1\}$, où $\mathcal{X}$ et $\mathcal{Y}$ sont, respectivement, les espaces d'entrée et de sortie de la tâche. La fonction apprise est alors

$$(h, r)(x) = \begin{cases} h(x) & \text{si } r(x) = 0, \\ ® & \text{sinon (r(x) = 1)}, \end{cases}$$

où ® désigne le refus de répondre.

Parmi les approches classiques du rejet, on peut distinguer deux grandes familles de méthodes. La première considère que h ne donne pas seulement une prédiction en $\mathcal{Y}$ mais fournit également une forme d'information de confiance, par exemple dans $\mathbb{R}$, qui peut être

utilisée pour effectuer un rejet en seuillant sur sa valeur [BW08]. C'est le cas, par exemple, de la régression logistique et des réseaux de neurones qui donnent une distribution de probabilité catégorielle en sortie, ou des SVMs qui fournissent une distance à la frontière de décision. Nous les qualifierons de méthodes *basées sur la confiance*. L'autre catégorie tente de mesurer les fluctuations du classifieur et favorise le rejet dans les zones de l'espace d'entrée où sa variabilité est la plus élevée. L'hypothèse avancée par ces méthodes est que le prédicteur est plus susceptible d'être incertain dans sa décision dans ces zones-là. Ces approches ont été particulièrement explorées dans le contexte de l'apprentissage actif [Set12] et dans les réseaux neuronaux bayésiens [GG16]. Nous les qualifierons de méthodes *basées sur la dispersion*.

Nous affirmons que, en fait, ces deux approches sont incomplètes dans le sens où elles ne saisissent que partiellement l'information sur l'incertitude. En effet, cette dernière peut être divisée en deux catégories : *l'ambiguïté de la tâche* et *l'incertitude du modèle*. La première est intrinsèque à la tâche que nous voulons accomplir. Par exemple, une image de la fleur d'une plante ne contient qu'une information partielle qui peut ne pas être suffisante pour distinguer deux espèces similaires. Cette incertitude provient du bruit de la mesure (prendre une photo dans notre exemple) ainsi que du bruit dans le processus d'annotation. Elle est, en fait, directement liée à la fonction de régression

$$\eta_k(x) = \Pr\left[Y = k \mid X = x\right]$$

de l'apprentissage avec un superviseur imparfait.

Cependant, ce n'est pas la seule incertitude qui apparaît quand on essaie d'apprendre un prédicteur. En effet, lorsqu'on nous donne un ensemble de données, nous devons faire face à l'incertitude du processus d'échantillonnage : nous n'avons accès qu'à une vue partielle de la distribution latente des données. Un échantillonnage différent des données nous induira à choisir un autre prédicteur. De plus, l'algorithme d'apprentissage peut être intrinsèquement stochastique. Ainsi, lors de l'apprentissage de réseaux neuronaux, en raison de la non-convexité de l'objectif d'apprentissage, la ré-exécution de la descente de gradient stochastique en utilisant une initialisation différente et un brassage différent des données d'apprentissage nous donnera une solution différente qui est souvent comparable en termes de précision tout en apportant de la diversité. Cette incertitude n'est pas intrinsèque à la tâche et survient lors de l'apprentissage du modèle, nous l'appellerons donc *incertitude du modèle*.

Ces deux types d'incertitude, l'ambiguïté des tâches et l'incertitude du modèle, saisissent des informations différentes et complémentaires. Nous proposons de les utiliser explicitement pour construire de nouveaux critères de rejet en essayant de les démêler avant de les fusionner de nouveau de manière adaptée.

Dans cet article, nous nous concentrerons principalement sur la classification binaire à l'aide de réseaux neuronaux. Nous présentons deux contributions principales. Premièrement, dans la Section 3, nous proposons de nouveaux critères de classification avec option de rejet en utilisant des mesures de l'ambiguïté des tâches et de l'incertitude du modèle. Deuxièmement, comme ces critères semblent avoir des performances similaires à celles des méthodes de référence dans nos expériences, dans la Section 4, nous montrons qu'il existe effectivement des informations supplémentaires à exploiter en démêlant les deux types d'incertitude présentés précédemment mais qu'elles peuvent être difficile à saisir sans apprendre un critère ad hoc sur un ensemble de validation.

2 Formulation du problème et état de l'art

La classification avec option de rejet a d'abord été introduite et étudiée dans [Cho57, Cho70] en utilisant des modèles probabilistes. Des travaux plus récents ont étudié ce problème dans le cadre de la théorie de l'apprentissage statistique en définissant une fonction de risque adaptée. Il est habituellement exprimé pour un coût de rejet donné de λ [HW06, CDM16] par

$$R_\lambda(h, r) = \mathbb{E}_{X,Y}\left[\delta_{h(X)\neq Y}(1 - r(X)) + \lambda r(X)\right] \quad (1)$$

et est minimisé pour le classifier de Bayes optimal (h^*, r^*) suivant :

$$\begin{aligned} h^*(x) &= \begin{cases} 1 & \text{si } \eta(x) \geq \frac{1}{2}, \\ 0 & \text{sinon,} \end{cases} \\ r^*(x) &= \begin{cases} 1 & \text{si } |\eta(x) - \frac{1}{2}| < \frac{1}{2} - \lambda, \\ 0 & \text{sinon.} \end{cases} \end{aligned} \quad (2)$$

Il s'agit donc d'un compromis entre le taux d'erreur et le taux de rejet pour ce coût λ et favorise le rejet dans les zones où l'ambiguïté de la tâche est plus grande.

Beaucoup de travaux théoriques partent de cette formulation et se concentrent principalement sur le cas binaire. Par exemple, [HW06] a étudié le taux de convergence des estimateurs plug-in et des minimiseurs du risque empirique. [BW08] a proposé une *hinge loss* pour les réjecteurs basés sur la confiance qui peut être

mise en œuvre dans la pratique et en propose l'étude théorique. [YW10] étudie d'autres fonctions objectives convexes tandis que [CDM16] étend cela à des réjecteurs d'une classe de fonctions différente de celle du prédicteur. Une autre approche consiste à apprendre séquentiellement des classifieurs qui peuvent rejeter avec un coût, une tâche appelée apprentissage séquentiel, comme étudié dans [TS13].

Parallèlement, [EYW10] a utilisé une stratégie basée sur les désaccords pour apprendre un prédicteur parfait dans le cas réalisable, c'est-à-dire en l'absence de bruit et lorsque le classifieur parfait est dans la classe des hypothèses. [WEY11, WEY15] ont ensuite étudié de telles approches pour le cas agnostique afin d'apprendre ce qu'ils appellent des classifieurs sélectifs point par point, c'est-à-dire que les prédicteurs ont le même taux d'erreur que celui optimal tout en essayant de maintenir le taux de rejet le plus bas possible. De plus, ils proposent de mesurer la performance de ces modèles, non seulement pour un coût de rejet λ donné, qui pourrait ne pas être quantifiable en pratique, mais sur l'ensemble de la courbe de risque/couverture, ce qui revient à comparer la performance pour tous les choix de λ en même temps [EYW10].

Cependant, ces travaux sont, soit purement théoriques, soit reposent sur une optimisation convexe. Cela peut être un problème pour les réseaux de neurones parce que leur processus d'apprentissage n'est pas encore entièrement compris. Ainsi, de nombreuses études empiriques et heuristiques ont été réalisées autour des critères de rejet et de l'estimation de l'incertitude pour ces modèles. Pour les réseaux neuronaux, les approches basées sur la confiance ont été étudiées depuis longtemps et continuent de l'être jusqu'à récemment dans [MW17, LPB17] par exemple. En ce qui concerne les méthodes fondées sur la dispersion, [GG16] interprète le *dropout* comme une approximation de l'inférence bayésienne et propose d'utiliser la variance du résultat comme mesure d'incertitude. Cependant, [GUEY19] montre que ces différentes mesures, basées sur la confiance ou sur la dispersion, peuvent être améliorées en choisissant, pendant la phase d'apprentissage, des modèles mieux adaptés à chaque zone de l'espace d'entrée. Plus largement, dans le contexte de l'apprentissage actif, des mesures basées sur la confiance et sur la dispersion ont été étudiées, [Set12] en propose une vue d'ensemble.

Les mesures de dispersions que nous introduisons dans ce papier sont basées sur le désaccord entre les modèles d'un ensemble. Cette notion de désaccord est importante dans l'apprentissage et a été exploitée à multiples reprises. En effet, elle a d'abord été utilisée pour obtenir un modèle plus performant que chacun des modèles individuels à travers le *bagging* [Bre96] et le *boosting* [SS99]. Certains travaux tels que [GLL$^+$15] fournissent une étude théorique de ces approaches. La notion de désaccord est également importante dans d'autres paradigmes d'apprentissage tels que l'apprentissage actif [H$^+$14] ou bien, comme développé précedémment, la classification avec rejet [WEY15].

Enfin, les différentes sources d'incertitudes sont généralement séparées en incertitudes *aléatoriques* et *épistémiques* [DKD09, KG17]. La première est définie comme la partie irréductible de l'incertitude, par opposition à la seconde qui peut être réduite en collectant plus de données ou en affinant le modèle. Cependant, comme cela est expliqué en détail dans [DKD09], la distinction précédente a un sens dans un modèle pour lequel il est explicite quelle incertitude peut être réduite. Nous préférons la terminologie de *l'ambiguïté de la tâche* et *l'incertitude du modèle* parce que nous ne considérons pas ici quelle incertitude peut être réduite et comment mais plutôt si elle est intrinsèque à la tâche ou si elle provient du processus d'apprentissage.

3 Exploiter les deux types d'incertitude

Dans cette section, nous étudions comment, à la fois, l'ambiguïté des tâches et l'incertitude du modèle peuvent être utilisées pour obtenir de nouveaux critères de classification avec option de rejet.

3.1 Incertitude dans le choix du prédicteur

En raison de la stochasticité de l'apprentissage et des processus d'échantillonnage des données, il y a une incertitude dans le choix de l'hypothèse h. Nous pouvons construire un critère basé sur la variabilité de la prédiction due à cette stochasticité.

Comme nous utilisons des classes d'hypothèses paramétrées, l'incertitude dans le choix de $h = h_\theta$ provient en fait d'une incertitude dans le choix des paramètres θ étant donné les données d'apprentissage $\mathcal{D}_{\text{train}}$, c'est-à-dire $\Pr\left[\theta = \theta' \mid \mathcal{D}_{\text{train}}\right]$. Ainsi, une première approche consiste à modéliser complètement cette distribution de probabilité sur les paramètres et une façon de le faire est d'utiliser des approches bayésiennes [Nea12]. Dans ce cas, une distribution *a priori* sur tous les paramètres, $\Pr\left[\theta = \theta'\right]$, est définie et sa la distribution *a posteriori* $\Pr\left[\theta = \theta' \mid \mathcal{D}_{\text{train}}\right]$ est calculée en appliquant la règle de Bayes.

Cependant, selon le modèle utilisé, il peut être difficile d'établir une distribution *a priori* adaptée, de plus,

le calcul de la distribution *a posteriori* peut s'avérer difficile. Il est possible d'utiliser des méthodes d'approximation mais, dans certaines situations, cela peut ne pas être suffisant. Cela est en particulier le cas pour les réseaux de neurones où l'application des méthodes bayésiennes est un domaine de recherche actif [Nea12, RMW14, BCKW15, GG16].

Néanmoins, comme dans le présent article, nous nous intéressons uniquement à la classification avec option de rejet, il n'est pas nécessaire de modéliser cette distribution complexe. Nous pouvons utiliser une approche plus directe pour estimer nos critères en supposant que nous pouvons échantillonner les modèles de notre distribution $\Pr[\theta = \theta' \mid \mathcal{D}_{\text{train}}]$. On peut y parvenir en simulant les différentes sources d'incertitude du modèle, c'est-à-dire

— la stochasticité de l'algorithme d'apprentissage en exécutant plusieurs fois l'apprentissage sur les mêmes données
— l'échantillonnage des données d'entraînement en utilisant des techniques de *bootstrap* et de sous-échantillonage pour simuler la stochasticité du processus de génération des données.

Une fois qu'il est supposé que nous pouvons échantillonner à partir de $\Pr[\theta = \theta' \mid \mathcal{D}_{\text{train}}]$, nous pouvons alors construire des critères basés sur la dispersion. En particulier, nous étudions dans la section suivante un critère basé sur le désaccord entre les prédicteurs de cette distribution.

3.2 Un critère pratique de désaccord

Les critères de désaccord ont été théoriquement étudiés dans le contexte de la classification avec option de rejet pour trouver un prédicteur ayant le même taux d'erreur que le classifieur parfait tout en ayant un taux de rejet faible [EYW10, WEY11, WEY15], ainsi que dans le cadre de l'apprentissage actif [H+14]. Les statégies proposés restent cependant théoriques et ne sont pas implémentables en pratique. Dans cette sous-section, nous proposons un moyen de les rendre pratiques et de les généraliser.

L'idée générale est de quantifier le désaccord entre les modèles $(h_\theta)_\theta$ d'un ensemble de modèles en regardant uniquement leurs prédictions $\hat{y} = h_\theta(x) \in \mathcal{Y}$. Pour cela, nous définissons la *mesure de désaccord* $\zeta(x)$ comme

$$\zeta(x) = \Pr\left[\hat{Y} = 1 \mid X = x\right] = \mathbb{E}_\theta\left[\delta_{h_\theta(x)=1}\right],$$

étant donné une mesure de probabilité sur l'ensemble des paramètres θ. Notez que cette quantité est différente de la fonction de régression $\eta(x)$ qui mesure l'ambiguïté de la tâche elle-même, $\Pr[Y = 1 \mid X = x]$, indépendamment des modèles. De plus, cette quantité n'est pas à proprement parler une probabilité de désaccord, mais elle capture cette information. En effet, le désaccord est maximal quand $\zeta(x)$ vaut 0.5 et minimal quand il vaut 0 ou 1.

Si nous modélisons complètement l'incertitude dans les paramètres $\Pr[\theta = \theta' \mid \mathcal{D}_{\text{train}}]$, il est alors possible de calculer le désaccord entre ces hypothèses en utilisant

$$\hat{\zeta}(x) = \int_{\theta'} \delta_{h_{\theta'}(x)=1} \Pr[\theta = \theta' \mid \mathcal{D}_{\text{train}}]\, d\theta'. \tag{3}$$

Cependant, si nous ne nous intéressons qu'à $\zeta(x)$, nous n'avons, en général, pas besoin de modéliser complètement la distribution sur les paramètres, ce qui peut être une tâche complexe comme le montre la sous-section précédente.

Au lieu de cela, parce que $\hat{y} \in \mathcal{Y}$ est discret, si on dispose de C échantillons, $\theta_1, \ldots, \theta_C$, nous pouvons estimer directement la distribution $\Pr\left[\hat{Y} = 1 | X = x\right]$ avec une approche fréquentiste :

$$\hat{\zeta}_{\text{freq}}(x) = \frac{1}{C}\sum_{i=1}^{C} \delta_{h_{\theta_i}(x)=1}. \tag{4}$$

Néanmoins, bien que cet estimateur soit non biaisé, il pourrait nécessiter beaucoup d'échantillons, soit un grand C, pour réduire sa variance.

En général, la plupart des modèles h_θ peuvent être décomposés en utilisant une fonction paramétrée $f_\theta : \mathcal{X} \to \mathbb{R}$ sur lequel est appliquée une fonction de décision $\delta_Z : \mathbb{R} \to \{0, 1\}$. Typiquement, $\delta_Z(z) = \delta_{z \geq 0}$. Lors de l'utilisation d'une fonction de coût logistique, z est appelé logit et nous pouvons même décomposer h un peu plus en appliquant, par-dessus f_θ, la fonction sigmoid $\sigma : \mathbb{R} \to [0, 1]$ pour transformer z en probabilité avant de prendre la décision qui devient $\delta_P(p) = \delta_{p \geq \frac{1}{2}}$. Ces décompositions peuvent être résumées comme suit :

$$\begin{aligned} x \xrightarrow{h_\theta} \hat{y} \quad &\Leftrightarrow \quad x \xrightarrow{f_\theta} z \xrightarrow{\delta_Z} \hat{y} \\ &\Leftrightarrow \quad x \xrightarrow{f_\theta} z \xrightarrow{\sigma} p \xrightarrow{\delta_P} \hat{y} \end{aligned}$$

Sur la base de la décomposition précédente, nous pouvons en fait utiliser une approche intermédiaire entre les deux extrêmes que sont les Équations (3) et (4). En effet, parce que $\hat{y}$ est une fonction (non paramétrique) de z et p, il suffit de modéliser l'incertitude dans l'espace des logits ou de la probabilité. A partir de ces distributions, nous pouvons ensuite dériver la mesure de désaccord en utilisant

$$\begin{aligned} \hat{\zeta}(x) &= \int_z \delta_{z \geq 0} \Pr[Z = z \mid X = x]\, dz \\ &= 1 - F_x^Z(0) \end{aligned} \tag{5}$$

et

$$\begin{aligned}\hat{\zeta}(x) &= \int_p \delta_{p\geq\frac{1}{2}} \Pr\left[P = p \mid X = x\right] dp \\ &= 1 - F_x^P(1/2),\end{aligned} \tag{6}$$

où F_x^Z et F_x^P sont respectivement la fonction de répartition de la probabilité conditionnelle dans l'espace des logits et des probabilités.

Cette méthode permet d'introduire des hypothèses et des connaissances à priori qui sont plus faciles à tester dans la pratique que la modélisation complète de la distribution des paramètres. En même temps, si ces hypothèses sont vérifiées, l'estimateur résultant convergerait plus rapidement que $\hat{\zeta}_{\text{freq}}(x)$, ce qui signifie que nous pourrions choisir un C plus petit, rendant cette approche plus pratique. Finalement, cette formalisation nous permet de mieux comprendre comment fusionner les différentes statistiques et moments, comme le montre la section suivante.

3.3 Choix de la distribution

Nous étudions maintenant quels sont les choix de distributions appropriées dans les espaces des logits et des probabilités.

En supposant qu'on nous donne plusieurs logits, $z_1, \ldots, z_C$, pour une entrée x, un choix courant est d'utiliser une loi normale pour modéliser leur distribution. Dans ce cas, nous pouvons ajuster les paramètres de la distribution en utilisant les estimateurs de maximum de vraisemblance habituels $\hat{\mu}_z = \frac{1}{C}\sum_{i=1}^{C} z_i$ et $\hat{\sigma}_z^2 = \frac{1}{C}\sum_{i=1}^{C}(z_i - \hat{\mu}_z)^2$. L'estimateur de l'Équation (5) devient alors

$$\hat{\zeta}_{\text{norm}}(x) = 1 - \phi\left(-\frac{\hat{\mu}_z}{\hat{\sigma}_z}\right),$$

où ϕ est la fonction de répartition de la distribution normale standard. Fait intéressant, ce critère est une fonction bijective de $\frac{\hat{\mu}_z}{\hat{\sigma}_z}$.

Alternativement, si nous considérons l'espace des distributions binaires p, un choix naturel est la loi bêta. Dans ce cas, il n'existe pas de forme close pour les estimateurs du maximum de vraisemblance de ses paramètres α et β. Il faut soit s'appuyer sur un algorithme itératif, soit utiliser la méthode des moments. Dans ce dernier cas, les estimateurs de la méthode des estimateurs sont égaux à $\hat{\alpha} = \hat{\mu}_p\left(\frac{\hat{\mu}_p(1-\hat{\mu}_p)}{\hat{v}_p} - 1\right)$ et $\hat{\beta} = (1-\hat{\mu}_p)\left(\frac{\hat{\mu}_p(1-\hat{\mu}_p)}{\hat{v}_p} - 1\right)$ avec $\hat{\mu}_p = \frac{1}{C}\sum_{i=1}^{C} p_i$ et $\hat{v}_p = \frac{1}{C}\sum_{i=1}^{C}(p_i - \hat{\mu}_p)^2$. L'estimateur de l'Équation (6) est alors égal à

$$\hat{\zeta}_{\text{beta}}(x) = 1 - I_{\frac{1}{2}}(\hat{\alpha}, \hat{\beta}),$$

où I_x est la fonction bêta incomplète régularisée.

Maintenant que nous disposons d'une mesure pratique de désaccord, nous pouvons établir un critère de classification avec une option de rejet telle que

$$c_{\text{disagree}}(x) = \max(\hat{\zeta}(x), 1 - \hat{\zeta}(x)). \tag{7}$$

Cependant, il ne s'agit là que d'une mesure de la dispersion des prédictions de l'ensemble, l'incorporation d'une mesure de l'ambiguïté de la tâche pourrait conduire à de meilleurs critères.

3.4 Critère de fusion

Si l'on examine la fonction de risque de la classification avec option de rejet de l'Équation (1), cette quantité est minimisée par le réjecteur optimal de Bayes de l'Équation (2). En estimant la fonction de régression $\eta(x)$, nous pouvons construire une règle *plug-in* qui nous permettrait d'effectuer un rejet basé sur notre estimateur $\hat{\eta}(x)$. Le taux de convergence théorique de cet estimateur *plug-in* a été étudié dans [HW06] où il a été démontré qu'il dépend de la qualité de l'estimateur $\hat{\eta}(x)$ et de la structure et du niveau de l'ambiguité $\eta(x)$. La construction d'un tel estimateur peut se faire en optimisant une fonction de coût *strictly proper* [GR07]. Il s'avère que la fonction de coût logistique est en fait *strictly proper*, un telle approche a par ailleurs été appliquée aux réseaux neuronaux dans [LPB17].

Cependant, cette méthode ne tient pas compte de la variabilité de l'estimateur qui change en fonction des zones de l'espace d'entrée. Cette variabilité pourrait donner lieu à une prédiction différente $\hat{y}$. En utilisant le critère de désaccord de la sous-section précédente, nous pouvons construire un nouveau critère en marginalisant notre incertitude dans le choix de la prédiction :

$$\begin{aligned}c_{\text{fusion}}(x) = &\Pr\left[\hat{Y} = 0 \mid X = x\right](1 - \hat{\eta}(x)) \\ &+ \Pr\left[\hat{Y} = 1 \mid X = x\right]\hat{\eta}(x).\end{aligned}$$

S'il n'y a pas de désaccord, c'est-à-dire si $\zeta(x) \in \{0, 1\}$, ce critère est simplement l'ambiguïté estimée de la prévision : $\max(\hat{\eta}(x), 1 - \hat{\eta}(x))$. De plus, pour une valeur égale de $\hat{\eta}(x)$, ce critère rejettera d'abord les zones de l'espace d'entrée où le désaccord est le plus fort. Cette quantité a donc des propriétés intéressantes.

En injectant notre estimateur de $\zeta(x)$ dans l'équation précédente, le critère devient

$$c_{\text{fusion}}(x) = (1 - \hat{\zeta}(x))(1 - \hat{\eta}(x)) + \hat{\zeta}(x)\hat{\eta}(x). \tag{8}$$

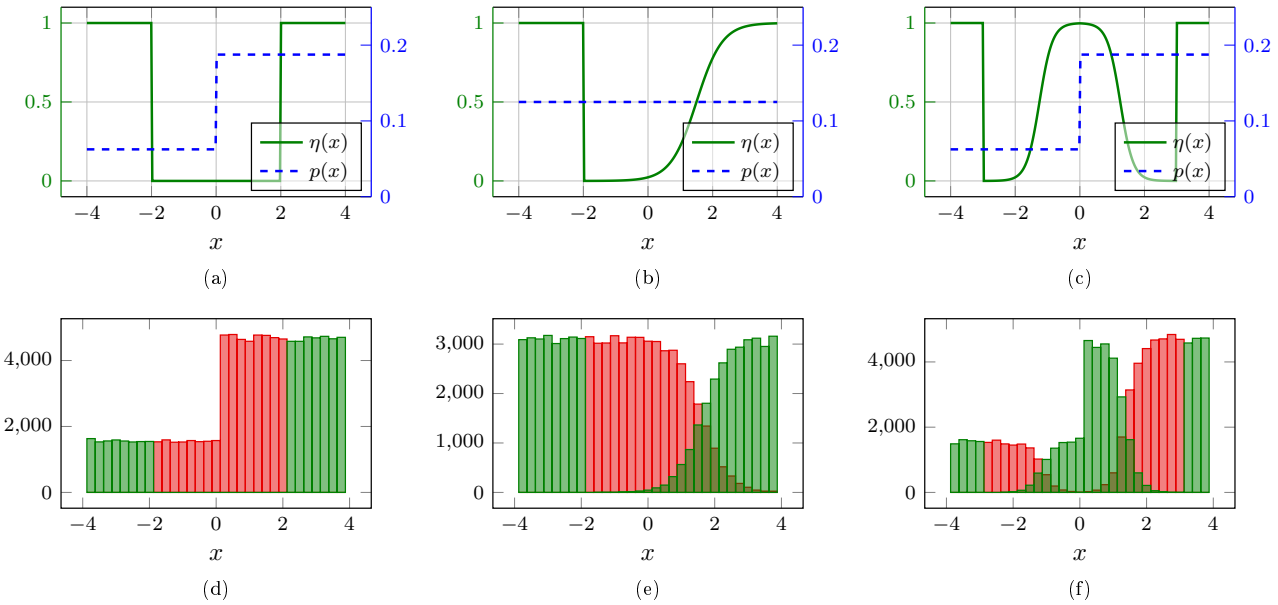


FIGURE 1 – Jeux de données synthétiques avec un degré variable d'ambiguïté et de densité des données. Sur la première ligne, la distribution de densité de probabilité de x est indiquée en pointiés bleus tandis que la fonction de régression $\eta(x)$ est indiquée en vert. Sur la seconde ligne, l'histogramme d'un échantillonage de cette distribution est tracé avec en vert les échantillons positifs et en rouge ceux négatifs.

3.5 Expériences synthétiques

Afin de comparer les différents critères de rejet, nous utilisons des jeux de données synthétiques où nous pouvons contrôler la quantité d'ambiguïté $\eta(x)$ et la densité des données $p(x)$. Nous utilisons trois jeux de données différents illustrés dans la Figure 1 :

— dans la Fig.1a, il n'y a pas d'ambiguïté, c'est-à-dire $\eta(x) \in \{0, 1\}$, mais il existe deux zones de densités uniformes différentes ;
— dans la Fig.1b, x est distribué uniformément mais il y a des zones d'ambiguïté de quantité différente ;
— dans la Fig.1c, c'est un mélange des deux scénarios précédents.

Ces jeux de données permettent de comprendre finement comment les prédictions du modèle et les critères de rejet se comportent sous différentes contraintes d'incertitude.

La mesure de performance que nous utilisons est la courbe RC [EYW10] et, en particulier, l'aire sous cette courbe telle que définie dans [GUEY19] que nous désignons RC-AUC. La courbe de risque/couverture (RC) est définie comme suit

$$R(h, r) = \mathbb{E}_{X,Y}\left[\delta_{h(X)\neq Y}\frac{1 - r(X)}{\phi(h, r)}\right],$$
$$\phi(h, r) = 1 - \mathbb{E}_X\left[r(X)\right].$$

Le risque R quantifie le taux d'erreur des échantillons qui ne sont pas rejetés. La couverture ϕ mesure le taux d'acceptation. Plus cette quantité est faible, meilleur est le compromis entre précision et taux de rejet.

Toutes les fonctions de rejet que nous considérons effectuent un seuillage se basant sur un critère c :

$$r(x) = \delta_{c(x)\geq\tau}. \tag{9}$$

Nous utilisons comme références les critères suivants :
— pour les approches basées sur la confiance : moyenne de $|\mu_p - \frac{1}{2}|$ dans l'espace de probabilité, notée "prob. mean", et moyenne de $|\mu_z|$ dans l'espace logit, notée "logit mean" [1]
— pour les approches basées sur la dispersion : la variance de $|\mu_p - \frac{1}{2}|$ notée "var. prob." et de $|\mu_z|$ notée "var. logit" et la moyenne de la divergence

1. ici, parce que nous considérons des tâches de classification binaire, les autres critères basés sur ces moyennes, par exemple, l'entropie etc, sont équivalents aux critères que nous utilisons

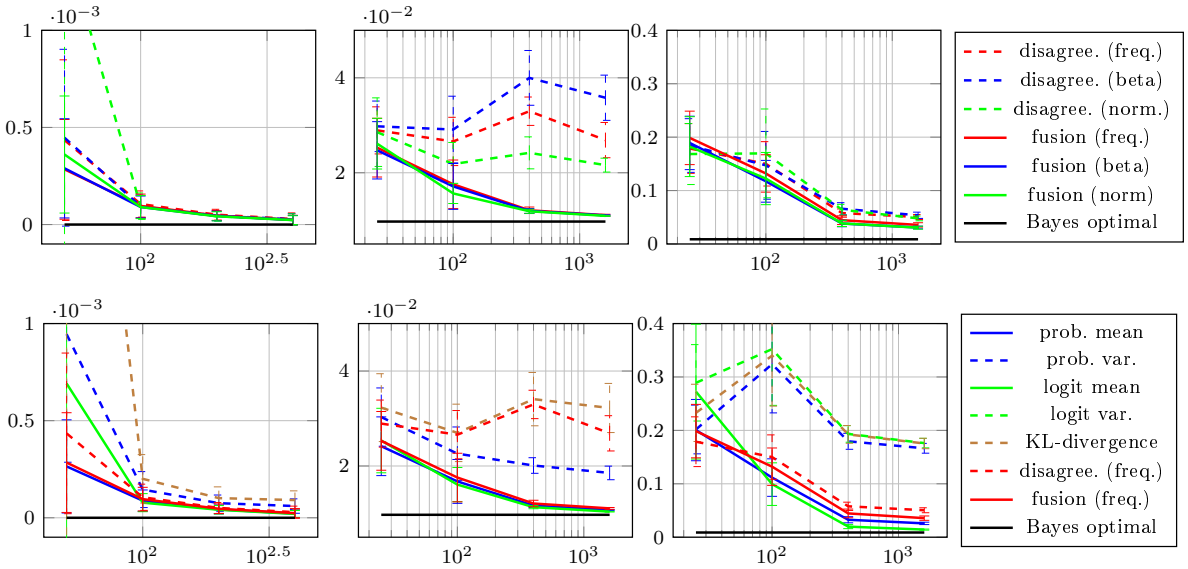


FIGURE 2 – Aire sous la courbe des courbes risk/couverture (RC-AUC) pour comparer les différents critères en fonction de la taille du jeu d'apprentissage. Les colonnes correspondent aux trois jeux de données synthétiques de la Figure 1. La première ligne compare les critères de désaccord et les critères de fusion entre eux. La seconde ligne les compare aux critères de base.

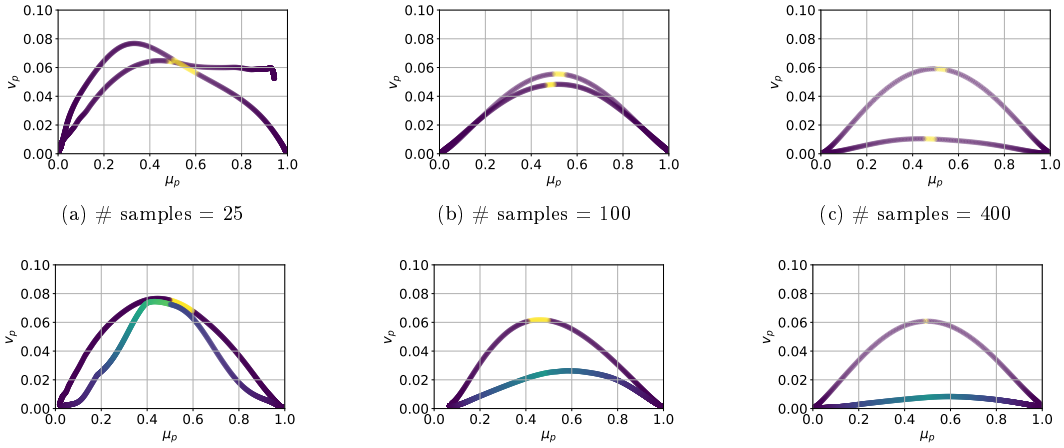
de Kullback-Leibler (KL) des prévisions des modèles individuels de l'ensemble comparé à la prévision moyenne telle que définie dans [Set12]

Notez que pour les critères basés sur la dispersion, c'est leur opposé qui est seuillé. Nous intégrons en outre le classifieur optimal de Bayes de l'Équation (2) pour fournir la plus faible valeur de la RC-AUC réalisable.

Nous comparons d'abord les différents critères basés sur les mesures de désaccord des Équations (7) et (8) dans la colonne de gauche de la Figure 2. Avec suffisamment de données, les critères de désaccord convergent vers le critère optimal en l'absence d'ambiguïté alors qu'en présence d'ambiguïté, ce n'est pas le cas. Cela est attendu car ces critères ne prennent pas en compte cette partie de l'incertitude, mais seulement le désaccord entre les modèles de l'ensemble. L'utilisation du critère de fusion pour incorporer l'ambiguïté à ces critères les améliore et les fait converger vers la performance du critère optimal. Parce que nous utilisons un ensemble de taille 1000, les approches fréquentistes peuvent être considérées comme les meilleurs estimateurs des critères de désaccord et de fusion. Les critères basés sur la loi bêta sont toujours assez proches et, par conséquent, cette dernière peut être considérée comme un bon choix pour modéliser la distribution des probabilités produites par l'ensemble. A contrario, les critères basés sur la loi normale ont une performance médiocre en l'absence d'ambiguïté tout en étant au même niveau ou légèrement meilleurs autrement. Cette loi ne semble donc pas parfaitement adaptée pour modéliser la distribution de l'espace des logits.

Lorsque ces critères sont comparés à ceux de référence, les critères de fusion donnent des résultats proches de la moyenne de la probabilité et se situent toujours à l'intérieur des fluctuations statistiques comme le montre la colonne de droite de la Figure 2. De plus, il est surprenant de constater que les autres critères basés sur la dispersion peuvent avoir de mauvais résultats et, dans certains cas, ils peuvent même ne pas apparaître sur le graphique.

Deux conclusions principales se dégagent de ces expériences. Premièrement, le bon comportement du critère fondé sur la moyenne de probabilité indique qu'il semble saisir une partie de l'incertitude du modèle, ce qui implique qu'il s'agit d'un estimateur biaisé en ce sens de $\eta(x)$. Par exemple, en l'absence d'ambiguïté, il est toujours performant alors que l'ambiguïté satisfait $\eta(x) \in \{0, 1\}$ et n'apporte donc aucune information sur l'incertitude dans ce cas. Deuxièmement, le critère de fusion proposé ne semble pas tenir compte



(a) # samples = 25 (b) # samples = 100 (c) # samples = 400

(d) # samples = 25 (e) # samples = 100 (f) # samples = 400

FIGURE 3 – La moyenne et la variance des prédictions dans l'espace de probabilité, (μ_p, v_p), pour chaque point de l'espace d'entrée, c'est-à-dire $-4 < x < 4$. La couleur de la courbe indique le taux d'erreur pour ces valeurs, le jaune correspondant à un taux d'erreur élevé et le mauve à un taux faible, le blue étant une valeur intermédiaire. Les première et deuxième lignes correspondent respectivement au premier et au deuxième jeu de données synthétique.

de l'incertitude supplémentaire, du moins dans le cadre de ces expériences synthétiques. Toutefois, dans la section suivante, nous analysons plus en détail ces jeux de données contrôlés afin de comprendre s'il existe effectivement des informations supplémentaires qui peuvent être exploitées et comment y parvenir.

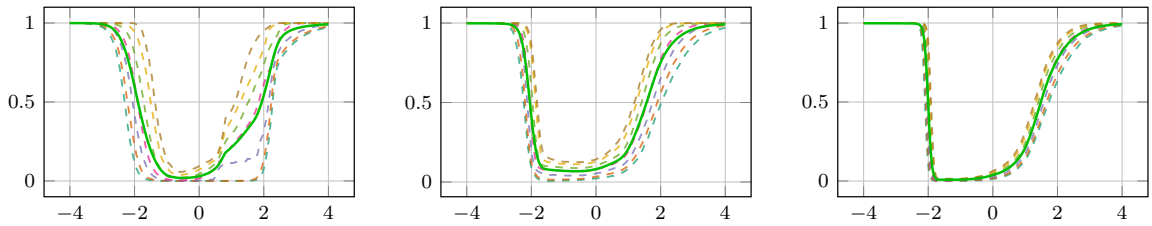
4 Étude approfondie des fonctions de rejet

4.1 Visualisation des frontières de décision

Les critères précédemment étudiés peuvent être considérés comme une simple décision prise par seuillage comme explicité dans l'Équation (9). La plupart d'entre eux définissent des frontières de décision, soit sur la moyenne et la variance dans l'espace de probabilité, (μ_p, v_p), soit sur la moyenne et l'écart-type dans l'espace logit, (μ_z, σ_z). Afin de visualiser la complexité de la frontière de décision idéale, la Figure 3 montre les valeurs de (μ_p, v_p) pour x variant sur l'espace d'entrée, de -4 à 4, pour les deux premiers jeux de données synthétiques. À chacun de ces points est associé la probabilité d'erreur en couleur.

Ces figures montrent que cette frontière de décision idéale peut en fait être assez complexe et varie en fonction de la taille du jeu d'apprentissage. De plus, la variance de la zone avec moins d'incertitude, c'est-à-dire pour x autour de -2, augmente avec le nombre d'échantillons d'apprentissage et cette zone finit par être celle avec la variance la plus élevée. Au premier abord, cela semble plutôt contre-intuitif. En effet, si l'on s'attend à ce que la variance donne directement l'information de l'incertitude du modèle, cette quantité devrait toujours diminuer au fur et à mesure que la taille des données d'entrainement augmente. De plus, il devrait être plus faible dans la zone de non-ambiguïté parce que le modèle devrait la capturer plus rapidement.

Si l'on examine de près la distribution des fonctions de prédiction des modèles dans l'ensemble de la Figure 4, cela est en fait logique. Dans les zones où l'ambiguïté est faible, lorsqu'on dispose de suffisamment de données d'apprentissage, l'incertitude de la fonction de prévision est faible et nous savons presque exactement où se produit le passage d'une classe à une autre. Cependant, comme tous les modèles de l'ensemble ont compris qu'il n'y a pas d'ambiguïté, chacun d'entre eux prédit ou bien 0, ou bien 1, avec une "confiance" élevée, ce qui entraîne une variance élevée de cette valeur de confiance à ce point précis tout en ayant un faible risque d'erreur. A contrario, dans les zones ambigües, la variance est plus faible car chaque modèle a connaissance de cette ambiguité mais est incertain sur l'endroit exact où la transition de la classe 0 à la classe 1 a lieu.



(a) # samples = 25 (b) # samples = 100 (c) # samples = 400

FIGURE 4 – Diagrammes en centiles des distributions binaires prédites par les modèles de l'ensemble pour plusieurs tailles du jeu d'apprentissage. Les centiles représentés en pointillés sont : 5%, 10%, 25%, 50%, 75%, 90% et 95%. La moyenne de ces prédictions est indiquée en vert.

La fonction de décision finale semble plus incertaine mais la variance calculée en un point reste relativement faible comparé à la zone non-ambigüe.

Les Figure 5 et Figure 6 montrent les frontières de décision des critères de rejet utilisés dans la section précédente, respectivement, dans l'espace de probabilité et l'espace logit. Les critères de désaccord et de fusion que nous proposons tiennent compte à la fois de la moyenne et de la variance dans ces espaces et sont donc plus susceptibles de rejeter des zones de forte variance même pour une moyenne constante ou le contraire. Cependant, ces graphiques mettent en lumière les limites des différents critères de base et des critères que nous proposons. En effet, aucun d'entre eux n'a une forme adaptée pour effectuer un rejet efficace sur les jeux de données synthétiques. Cela est attendu, car plus la variance est élevée, plus il est probable que nous rejettions. Mais cette hypothèse n'est pas la bonne ici, ce qui souligne la complexité de l'élaboration d'un critère théorique. Cependant, comme on peut le remarquer dans la Figure 3, les zones de taux d'erreur différents peuvent toujours être séparées dans l'espace de (μ_p, v_p). Nous pouvons donc essayer d'apprendre de manière supervisée à mieux rejeter.

4.2 Apprentissage supervisé du réjecteur

Étant donnée que la frontière de décision du réjecteur est une fonction complexe, comme nous l'avons vu dans la sous-section précédente, nous proposons d'essayer de l'apprendre à partir des données en utilisant un jeu de données de calibration. Pour ce faire, nous devons revenir à la fonction de risque de la classification avec option de rejet de l'Équation (1). Dans notre cas, h est déjà appris et il suffit donc d'optimiser le risque précédent par rapport à r. Ce scénario a été étudié dans le contexte de l'apprentissage séquentiel [TS13]. Cela revient simplement à apprendre une tâche de classification binaire pondérée

$$R_\lambda^{\text{rej}}(h, r) = \mathbb{E}_{X,E}\left[w_E \delta_{r(X)\neq E}\right], \tag{10}$$

où E est la variable aléatoire correspondant à une erreur de prédiction, $E = \delta_{h(X)\neq Y}$, et les poids sont égaux à $w_0 = \lambda$ et $w_1 = 1 - \lambda$.

Le réjecteur de Bayes optimal est dans ce cas égal à

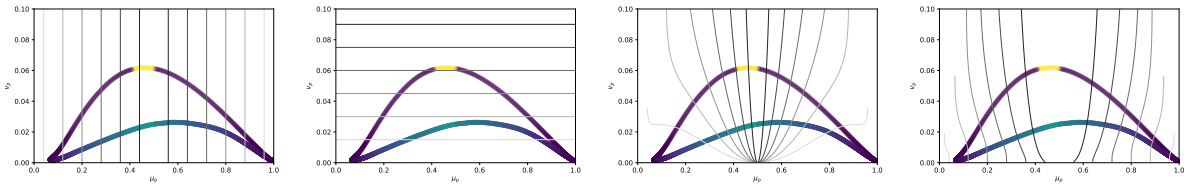
$$r^*(x) = \begin{cases} +1 & \text{si } \eta_E(x) > \lambda, \\ 0 & \text{sinon,} \end{cases} \tag{11}$$

où $\eta_E(x)$ est la fonction de régression de l'Équation (10), c'est-à-dire $\eta_E(x) = \Pr\left[Y \neq h(X) \mid X = x\right]$.

Nous pouvons alors apprendre n'importe quel modèle de classification habituel pour effectuer cette tâche. En faisant varier la valeur de λ, on peut alors reconstituer la courbe RC en réapprenant le réjecteur adapté à chaque fois.

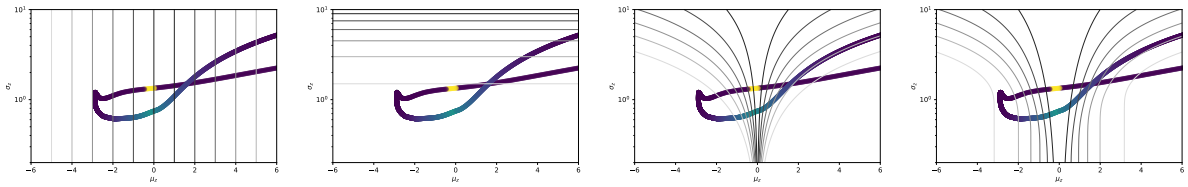
Notre but ici n'est pas de construire le meilleur modèle de rejet mais plutôt de montrer qu'en changeant notre espace d'entrée pour le rejet de $x \in \mathcal{X}$ à $(\mu_p, v_p) \in \mathbb{R}^2$, nous conservons encore suffisamment d'informations discriminantes pour effectuer le rejet. Nous ne prétendons pas qu'il s'agit du meilleur espace pour accomplir cette tâche mais qu'il est suffisant pour apprendre de meilleurs critères de rejet que les critères précédents.

Nous utilisons des classifieurs polynomiaux de degré 3 entrainés avec une fonction de coût logistique. Nous utilisons beaucoup de données d'étalonnage, soit 1000 échantillons, pour apprendre le réjecteur. Ce chiffre n'est pas réaliste dans un scénario réel où nous préférerions utiliser ces données pour améliorer notre prédicteur. Cependant, notre but ici est de comprendre à quel point un dispositif de rejet peut devenir meilleur si nous lui permettons d'être plus complexe et d'être



(a) moyenne des probabilités (b) variance des probabilités (c) désaccord (bêta) (d) fusion (bêta)

FIGURE 5 – Frontières de décision des différents critères dans l'espace (μ_p, v_p) de l'espace de probabilité.



(a) moyenne des logits (b) écart-type des logits (c) désaccord (normale) (d) fusion (normale)

FIGURE 6 – Frontières de décision des différents critères dans l'espace (μ_z, σ_z) de l'espace des logits.

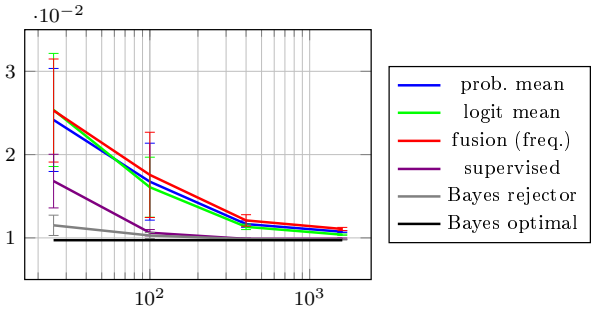


FIGURE 7 – Aire sous la courbe des courbes RC (RC-AUC) comparant le réjecteur supervisé aux différents autres critères avec une taille du jeu d'apprentissage variable sur le deuxième jeu de données synthétique.

dépendant de la tâche. La Figure 7 montre la courbe RC-AUC du rejecteur par rapport aux critères précedents et aux réjecteurs de Bayes des Équations (11) et (2) sur le deuxième jeu de données synthétique. La fonction de rejet apprise est en effet bien meilleure que les critères précédents. Cela montre la richesse de l'information contenue dans le plan (μ_p, v_p).

5 Conclusion et travaux futurs

Les critères classiques, tels que la valeur maximale de la probabilité prédite par un réseau neuronal, fonctionnent bien dans la pratique, cependant ils n'utilisent pas toute l'information d'incertitude disponible. L'analyse des prédictions d'un ensemble met en lumière certains comportements contre-intuitifs qui soulignent notre mauvaise compréhension de l'information d'incertitude capturée par ces modèles. Trouver un bon critère de rejet qui tire parti de toute cette information et qui généralise à différentes tâches est un problème difficile.

Une suite naturelle de ces travaux est d'effectuer des analyses sur des données réelles afin de vérifier que les comportements présentés ici se produisent également dans des jeux de données usuels. De plus, ce travail ouvre la voie à l'élaboration d'un meilleur critère de rejet. Une piste de recherche en ce sens consiste à trouver un espace de représentation plus adapté pour l'apprentissage supervisé des réjeteurs afin de permettre à ces derniers d'être calibrés en utilisant beaucoup moins de données.

Remerciements

Ce travail a été partiellement financé par le projet ANR WeedElec.

Références

[BCKW15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International*

Conference on Machine Learning, pages 1613–1622, 2015.

[Bre96] Leo Breiman. Bagging predictors. *Machine learning*, 24(2) :123–140, 1996.

[BW08] Peter L Bartlett and Marten H Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(Aug) :1823–1840, 2008.

[CDM16] Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with rejection. In *Lecture Notes in Computer Science*, pages 67–82. Springer International Publishing, 2016.

[Cho57] C. K. Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, EC-6(4) :247–254, 1957.

[Cho70] C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1) :41–46, 1970.

[DKD09] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic ? does it matter ? *Structural Safety*, 31(2) :105–112, 2009.

[EYW10] Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(May) :1605–1641, 2010.

[GG16] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation : Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059. PMLR, 2016.

[GLL+15] Pascal Germain, Alexandre Lacasse, Francois Laviolette, Mario Marchand, and Jean-Francis Roy. Risk bounds for the majority vote : From a pac-bayesian analysis to a learning algorithm. *The Journal of Machine Learning Research*, 16(1) :787–860, 2015.

[GR07] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477) :359–378, 2007.

[GUEY19] Yonatan Geifman, Guy Uziel, and Ran El-Yaniv. Bias-reduced uncertainty estimation for deep neural classifiers. In *International Conference on Learning Representations*, 2019.

[H+14] Steve Hanneke et al. Theory of disagreement-based active learning. *Foundations and Trends® in Machine Learning*, 7(2-3) :131–309, 2014.

[HW06] Radu Herbei and Marten H Wegkamp. Classification with reject option. *Canadian Journal of Statistics*, 34(4) :709–721, 2006.

[KG17] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision ? In *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017.

[LPB17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[MW17] Amit Mandelbaum and Daphna Weinshall. Distance-based confidence score for neural network classifiers. *arXiv preprint arXiv :1709.09844*, 2017.

[Nea12] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 1278–1286, 2014.

[Set12] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1) :1–114, 2012.

[SS99] Robert E Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3) :297–336, 1999.

[TS13] Kirill Trapeznikov and Venkatesh Saligrama. Supervised sequential classification under budget constraints. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 581–589. PMLR, 2013.

[WEY11] Yair Wiener and Ran El-Yaniv. Agnostic selective classification. In *Advances in neural information processing systems*, pages 1665–1673, 2011.

[WEY15] Yair Wiener and Ran El-Yaniv. Agnostic pointwise-competitive selective classification. *Journal of Artificial Intelligence Research*, 52 :171–201, 2015.

[YW10] Ming Yuan and Marten Wegkamp. Classification methods with reject option based on convex risk minimization. *Journal of Machine Learning Research*, 11(Jan) :111–130, 2010.

Joint Label/Example Hyperbolic Representation for Extreme Classification

Thomas Gerald and Nicolas Baskiotis

Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Abstract

Extreme multi-label classification task deals with nowadays problems involving up to millions of labels. The extremely large number of labels requires efficient methods not only in terms of prediction performances but also in terms of time processing and memory management. Most recent works focus on representation learning approaches: embeddings allow to infer a semantic structured space with interesting generalization properties; the label prediction is often performed by using approximated nearest neighbors search. Recent works in Language Modeling and especially in Question Answering have shown increasing performances using the hyperbolic space to learn the representation. The Poincaré ball model is indeed more relevant to represent ontology than the euclidean space. In this work, we propose to explore representation learning in hyperbolic space in the context of extreme classification. The proposed model performs a joint embedding of examples and labels. Most of the embedding methods for extreme classification only learn example embedding. Thus in order to structure the space, they require to precompute examples neighborhood. In our approach, learning a joint embedding allows structuring the space without any expensive preprocessing. Experiments conducted on real-world datasets show the performances of our model compared to state of the art.

1 Introduction

Large-Scale Multi-Label classification also known as *Extreme Classification* has become a real challenge of interest this last decade. Its objective is to develop classification algorithms able to identify multiple relevant labels among a very large label set for an example given as input. Large-scale classification is different due to the large label and features dimension, especially regarding the number of labels to discriminate. Assigning automatically topics/labels for a large amount of data collected by social network or encyclopedia is a main subject of interest. Even in the industry such as e-commerce platforms, finding relevant object associated with a query is an important task and requires often automatic tagging. Recently the domain has grown significantly and most leading methods are based on representation learning approaches. Finding relevant representation is, therefore, a key task to tackle *extreme classification*. In extreme classification leading methods are often based on tree involving to discover latent hierarchy. Representation models do not include this hierarchical information, however recent works on embedding hierarchy using hyperbolic representation space have proven their efficiency to capture the latent hierarchical information.

In this work, we propose to investigate representation learning for *Extreme Multi-Label Classification* by using instead of the usual Euclidean metric space the Poincaré ball model. To structure the embedding space we learn a joint label/example representation. We study the structure of the learned space and show experimentally that it is a relevant alternative to the euclidean space for multi-label classification. We present experiments on several large scale datasets and compare our approach to state of the art models. We show that for certain types of corpus our approach is relevant, especially when the number of dimensions of the embedding space is low.

Classification of documents, images or items has raised interest during these previous years due to the spread of available data, the diversity of new methods and the growing of computational power. Document classification has benefited from many improvements reaching impressive performances. However, classification of documents can be inefficient in computation time, prediction performances and memory management when the number of labels to discriminate be-

come too large. To address this issue, the Extreme Multi-label Classification domain is an active research field to deal with the extremely large number of classes (from 10^4 up to 10^7).

Algorithms for *Extreme multi-label classification* follow mainly three paradigms: 1) One Versus Some/All paradigm (OVA); 2) Hierarchical paradigm using classifiers organized by a graph structure; 3) Representation learning paradigm.

Historical baselines dealing with multi-label setting are mainly based on the *One versus All* paradigm: for each label present in the dataset a classifier is learned to distinguish it from the others; the prediction step infers the relevant labels using a voting scheme mixing the outputs of all the classifiers. The approach is extremely costly in terms of computation time and in memory storage, needing an extremely large number of classifiers growing linearly with respect to the number of labels. Another major drawback of this paradigm is its incapacity to exploit label correlation [ZLLG18]. To reduce the computation cost, methods using sparsity have been proposed such as `PDSparse` [YHZ+, YHD+17] using weight regularization to ensure the sparsity of each classifier [ZH05]. With the growth of parallel computational power, distributed `OVA` methods `DiSmec` success to learn classifiers efficiently using 1000 CPU cores, and manage the memory/time tradeoff by integrating pruning methods.

Hierarchical approaches exploit the divide and conquer paradigm to reduce drastically the number of classifiers used to infer labels. The classifiers are embedded in a tree structure, where each split corresponds to a split of the label set. The classifier attached to a given node is fitted in order to recognize at which child label subset belongs an example. Each leaf of the tree corresponds ideally to a unique label. The classification process begins at the root note and follows a path through the tree until a leaf is reached. The complexity of the inference is thus logarithmic with respect to the number of classes. Although finding an optimal tree structure and label organization is a complex problem, the recent approaches `FastXML` and `PfastreXML`[PV14, JPV16] reach the state of the art performances. *FastXML* recursively partitioning nodes maximizing the *Normalized Discounted Cumulative Gain* (`nDCG`) ranking loss; *PFastreXML* uses for the partitioning a new loss function, the *Propensity scored Loss*. This loss is also used to evaluate performances taking into account the frequencies of labels in corpora. Still using the divide and conquer paradigm, the recent method `PARABEL`[PKH+18] merges OVA and hierarchical approach proposing overlapping label partitioning at each node.

1.1 Learning embedding for extreme classification

Label dimension reduction approaches reduce the complexity of the inference step by taking into account the label correlations.

A first approach consists to factorize the label matrix, the label matrix being the matrix containing for each example a label vector with value 1 if labels are annotated with. Remarking that labels can be correlated led to find a way to factorize the matrix. Thus, the labels matrix can be embedded in a sub-dimensional space. This hypothesis is named the low-rank assumption, it has been applied in [HKLZ09, TL12] by factorizing the matrix using eigenvectors with the largest eigenvalues. Similarly, methods have been proposed to reduce the feature space dimension [CL12]. However, in the case of extreme classification, the low-rank assumption is very naive: in case of large diversity, labels are represented only a few times in corpora [Tag17]. Despite the dimensionality reduction, these methods are still time costly especially due to the decoding process within a time complexity still linear in the number of labels.

In order to deal with a large number of labels during the prediction process, several methods have proposed to use nearest neighbors search procedure. The number of label candidates is thus drastically reduced as only the labels of nearest examples are taken into account. For instance, the *SLEEC* approach [BJK+15] operates first partitioning of the input space and then learns mapping features to embeddings in each partition according to a label/example similarity. The inference is finally done by retrieving the labels of the nearest neighbors of the input example. This method has impressive performances on many corpora. The more recent method *AnnexML*[Tag17] proposes several improvements, particularly, it introduces a new partitioning scheme taking into account the label similarity, a new loss function to define the similarity between examples and an ensemble learning scheme to improve performances. The inference time is also drastically lowered thanks to an approximate nearest neighbor search.

1.2 Hyperbolic representation space

Using hyperbolic geometry to embed hierarchical information has recently raised the interest of the machine learning community.

Particularly the Poincaré ball model having good properties such as differentiable distance or hierarchical compliant geometry. The point at the origin of the ball is thus a natural candidate to represent the top hierarchy and the edge of the ball the leaves. Several recent works use this space: [NK17] have proposed an embedding of word vectors with most general concepts closer to the center of the ball;

Ganea et al[SDSGR18] have shown the efficiency of the representation space in extremely low dimension, producing state of the art results for *WordNet* embeddings using only two-dimensional latent space, but with complete hierarchy known. Moreover, it has been empirically proven to be efficient in not Tree like corpora. If previous methods are directly learning to embed without parametric functions, learning a mapping function from a features space has also been studied.

The algorithm *HyperQA* [TLH17] have reached state of the art on Question-Answering domain by using neural-networks to produce Poincaré Ball embedding.

They also empirically show that using Poincaré ball model for information retrieval through parametric function can lead to state of the art performances and confirm the low dimension efficiency. More recently the works *Hyperbolic Neural Networks*[GBH18] provided a set of neural layers for deep learning approaches using hyperbolic structure.

If learning representation for large scale multi-label classification remains a relevant approach, capture the hierarchical structure of data is not covered by today *Extreme classification* representation methods. Indeed, it is difficult to observe and embed hierarchy in Euclidean space while the majority of large scale corpora have a latent taxonomy. Thus using Hyperbolic representation space could lead to disambiguate example/labels hierarchy, where Euclidean can not. Moreover, due to the robustness of those model using low dimension, nearest neighbors search prediction could be more stable using a lower dimension. Thus leading to obtaining the same performances faster because of lower dimensionality.

2 Proposed approach

In this work we propose to study a model mapping the *feature* space and the *labels* into a joint hyperbolic representation space. Thus we design two functions, one for the feature embedding and the other one for the label embedding. In order to learn these representations jointly, we design a loss that tends to set closer the embedding of an example and the embedding of a label when the example belongs to the given label. Once representation learned examples representation within similar labels tends to be close, thus we perform prediction by annotating labels of an example according to its neighbors in the representation space. The section is organized as follows: first we introduce the notation being used; next we present a quick introduction to the *Poincaré Ball* model and give insight about the geometry; the model architecture and the associated learning procedure are next presented; finally, we present the inference step.

2.1 Notations

Let $\mathbf{X} \in \mathbb{R}^{N \times m}$ be the data matrix containing N examples lying in a m dimensional feature space; $x_i \in \mathbb{R}^{\mathbb{N}}$ the i-th example; $\mathbf{Y} \in \{0,1\}^{N \times L}$ the associated label matrix with L labels, such that $y_{i,k} = 1$ if the example i is annotate with label k (otherwise $y_{i,k} = 0$). We will note y_i the i-th line of the label matrix, corresponding to the label vector of the example x_i. We aim to embed both examples x_i and labels $k \in \{1, 2, 3, \ldots, L\}$ in the Poincaré ball model.

2.2 Poincaré Ball Model

The Poincaré ball model is a model of hyperbolic geometry lying in an n-dimensional sphere. It can be intuitively seen as the projection of a hyperbolic function of dimension $n+1$ to a ball of dimension n. The original model maps points of the hyperbolic function in the open unit ball. A particularity of this space is that the norm of a vector tends to infinity when the vector tends to the boundary of the ball. Thus a point near the ball center tends to be closer to all points than points near the boundary. This is explained by the formulation of the metric in the space. This property makes the Poincaré ball model a relevant choice for embedding hierarchical data by mapping the nodes near the root closer to the origin and leaves near the boundary, as represented in the figure 2.2. Each segment in the figure has the same length in the hyperbolic space.

Formally, let be $\mathcal{B}^n = \{x \in \mathbb{R}^n | \, \|x\| < 1\}$ the n-dimensional open unit ball. We refer now as $\mathbb{H}_n$ the Poincaré ball model of dimension n corresponding to the metric space on $\mathcal{B}^n$ equipped with the metric g_x, i.e $(\mathcal{B}^n, g_x)$. The g_x metric is defined as (with g^E the euclidean metric tensor):

$$g_x = \left(\frac{2}{1 - \|x\|_2^2} \right)^2 g^E$$

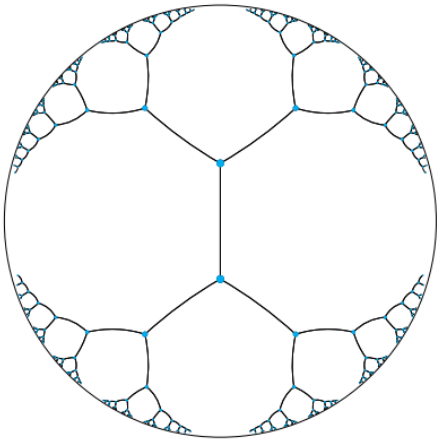


Figure 1: Embedding of a tree in $\mathcal{B}^2$ (This figure come from the paper [NK17])

The definition of the hyperbolic metric space depicts that a point will have a hyperbolic norm depending on the inverse of the square of the Euclidean norm: embedding close to the boundary of the ball will have a norm tending to infinity. The hyperbolic distance in the *Poincaré ball* model d is defined for two vectors $x_1, x_2 \in \mathbb{H}_n$ as following :

$$d(x_1, x_2) = arcosh\left(1 + 2\frac{\|x_1 - x_2\|_2^2}{(1 - \|x_1\|_2^2)(1 - \|y_1\|_2^2)}\right)$$

The distance is differentiable, thus usual back-propagation algorithms can be used while taking into account the curvature of the space into the optimization procedure.

2.3 Embedding into the Poincaré ball model

Our objective is to embed in the same space labels and examples in hyperbolic space in such a way that examples with similar annotation being close to each other. As there is no mapping function to learn for the labels, we represent their embeddings by a matrix $W^l \in \mathbb{R}^{L \times n}$, with L the number of labels and n the dimension of the Poincaré ball, i.e. the size of the latent representation. The i-th line of the matrix W^l, $l_i \in \mathbb{R}^n$, correspond to embedding of the i-th label . In order to learn the mapping between the feature space and the hyperbolic space, we use a simple linear projection layer $f_\theta : \mathbb{R}^M \to \mathbb{R}^n$ with $\theta \in \mathbb{R}^{M \times n}$. Let r_j the projection of the example j obtained by the function f_θ:

$$r_j = f_\theta(x_j) = \theta' \cdot x_j$$

More complex mapping functions can be used (for instance neural networks with multiple layers) but experiments showed that a linear projection is sufficient to reach the best performances.

In order to constrain the projection of the labels and the examples to the n-dimensional unit ball, we define the following projection:

$$p(x) = \begin{cases} \frac{x}{\|x\|+\varepsilon} & \text{if } \|x\| \geq 1 \\ x & \text{otherwise} \end{cases}$$

Thus the label for the i-th label is given by $p(l_i)$ and embedding of an example x_j by $p(f_\theta(x_j)) = p(r_j)$.

2.4 Loss function

Most of recent contributions in representation learning for extreme classification are trying to obtain close example embeddings for similar examples according to their label vectors. For instance, *AnnexML* [Tag17] proposes a similarity between examples based on the cosine similarity of the vector labels:

$$cos(y_i, y_j) = \frac{y_i \cdot y_j}{\|y_i\|_2 \|y_j\|_2}$$

with y_i the label vector of x_i. Today approaches to structure the embedding space is to first for each example define a set of neighbors based on a label similarity and then design a cost function setting closer examples within the same neighborhood.

The following loss is used in [Tag17] for a given example x_j and a example x_v in the neighborhood V_j of x_j (i.e. the k-nearest neighbors of the example x_j with respect to the cosine similarity between the label vectors), and r_i the euclidean embedding associate to the example x_i:

$$L(x_j, x_v) = -log(\frac{e^{\alpha cos(r_j, r_v)}}{\left[\sum_{k \notin V_j} e^{\alpha cos(r_j, r_k)}\right] + e^{\alpha cos(r_j, r_v)}})$$

With $k \notin V_j$ denote examples x_k who do not belong to the x_j neighborhood and α controlling the smoothness of the softmax loss.

This loss model the negative log-probability of having embedding of x_v belonging to the neighborhood of x_j embedding.

This method requires before the learning step to process the dataset using nearest neighbors search in order to get for each example their neighborhood.

Contrary to the state of the art methods, we propose to learn the example embeddings based on the label embedding, by setting closer example embeddings using label representation. This is made possible because both representations lying in the same space. Thus we do not need to pre-compute any label correlations or pre-computing neighborhood for the examples. The loss proposed in this work is similar to the previous one, but instead of processing distance between the example representations, we compute the distance of the representation of an example to the representation of one of its labels. Let $V(j) = \{k|y_{jk} = 1\}$ the labels of the example x_j; the proposed loss for an example x_j and a label i such that $i \in V(j)$ is :

$$L(x_j, l_i) = -log(\frac{e^{-\alpha d(r_j, l_i)}}{\sum\limits_{k \notin V(j)} e^{-\alpha d(r_j, p(l_k))} + e^{-\alpha d(r_j, p(l_i))}})$$

with $r_j = p(f_\theta(x_j))$ the projection of x_j and l_i the representation of the label i. Similarly, this can be interpreted as the negative log-probability of having the label i rather than another for the example x_j. When the convergence is achieved the label space has embedded closer examples with shared labels, and the representation of the examples should be close to the representation of their labels. Moreover, we expect that examples having the most label in common will be rather close to each other in terms of hyperbolic distance. In practice, for an example x_j, we use negative sampling by randomly sampling labels rather than considering all labels which are not in $V(j)$. Because of the large number of labels, there is only a low probability to sample a label belonging to $V(j)$. Due to the structure of the hyperbolic representation space, rare labels should be close to the boundary of the ball. At the opposite, top labels appearing often in the dataset should closer to the origin.

2.5 Optimization algorithm

All the presented function are differentiable, however, we can not learn that representation using regular gradient descent algorithm due to the hyperbolic embedding: the gradient depends on the metric g_x. In order to take into account the slope of the space curvature, the paper [NK17] has proposed a gradient descent algorithm named projected *RSGD* which scale the gradient according to the metric of the space. To update the label embeddings the gradient obtained by backpropagation is scaled by the inverse of the metric g_x^{-1}. Considering at a step t of the algorithm the representation vector l_i^t of the i-th label, the update of the representation at $t+1$ is given by :

$$l_i^{t+1} = l_i^t - \eta \left(\frac{1 - \|l_i^t\|_2^2}{2} \right)^2 \nabla_{p(l_i)} L(x_i, l_i)$$

with ϵ the learning rate. In a similar way, the update of the parameter θ^t of the mapping function of the examples is given by the following rule :

$$\theta^{t+1} = \theta^t - \eta \nabla_{\theta^t}(r_j) \left(\frac{1 - \|r_j\|_2^2}{2} \right)^2 \nabla_{r_j} L(x_j, l_i)$$

Experimentally we obtained better results using the *Adam* optimization algorithm applying the previous inverse norm factor on the value to update, however, due to exponentials decays in *Adam* we can not consider this method as formal.

2.6 Inference step

To annotate a new example x we embed the example to the representation space with $p(f_\theta(x))$. Then an *approximate K Nearest Neighbor Search* according to the distance d is performed to retrieve closest training examples. The number of occurrences of each label associate to embedding within the neighborhood of x representation is aggregated and ranked accordingly. The most relevant labels correspond to the most occurring labels in the neighborhood, with the highest apparition frequency.

To perform the approximate neighbor search, we experimentally observe that random selection of centroids among the representations of the training examples to construct the clusters does not decrease performances in comparison to exact neighbors search.

The approximate search allows to largely decrease the inference time. In the remainder of this paper, we will use $\frac{N}{3000}$ random clusters by default.

3 Experiments

3.1 Experimental settings

Table 1 presents the selected datasets to evaluate the proposed method. All of them are real-world corpora based on Wikipedia annotations (*Wiki10-31K, WikiLSHTC-325K*), web page annotations (*Delicious-200K*) or web market items/query annotations (*AmazonCat-13K, Amazon-670K*). The datasets have major differences: the Wikipedia and the web page description dataset have a hierarchical label organization while the two amazon datasets have a more flat label organization; the number of labels

by example and the number of examples for a given label also differ from a factor larger than 10 between the datasets. Moreover, corpus as *wiki10* contains very few examples compared to the number of labels, thus the number of labels will be rarely considered during learning. At the contrary in the case of *AmazonCat* where the number of examples per label is large, most of the labels will be updated many times.

In the following experiments, when not specified the selected dimension of the representation space is generally fixed to 50 (like in most multi-label classification works [BJK+15, Tag17]). A validation set is used to control the overfitting, build by sampling 5% of the training set. The learning process is stopped when precision does not increase during 25 epochs. The label embedding matrix is initialized according to a normal distribution $\mathcal{N}(0, 1e-3)$ and the mapping function *f_theta* with the distribution $\mathcal{N}(0, 1e-4)$.

Concerning the optimization, the *adam* optimizer [KB14] is used with hyperbolic Poincaré retraction (see 2.5), which shown experimentally better results than the regular stochastic descent. The main evaluation metric considered is the *precision at k* (p@k): given $\hat{y}_i$ the ordered label prediction inferred for an example x_i, with $\hat{y}_i^j \in \{1, 2, \ldots, L\}$ the j-th predicted label, the precision is computed as follows:

$$p@k = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{k} \sum_{j=1}^{k} y_{i,\hat{y}_i^k}$$

Table 2 presents a naive baseline inspired by [Tag17] showing the difference between the datasets regarding the label distribution. The baseline named *Most-Common*, predicts the same ordered label vector for all the examples, sorted from the most frequent label in the training dataset to the less frequent one (e.g in *wiki10*, the five most common labels are *wikipedia, wiki, reference, history, research*).

3.2 Evaluation

Table 3 shows the results of the proposed approach (*Hyp*) compared to the state of the art results. Outside our results, the reported results are from the *Extreme classification repository*[1] except for the *AnnexML* algorithm for which the results come from [Tag17]. Following their experimental settings, the predictions are made by an ensemble learning model, averaging the vote of 15 different models. The performances differ widely with respect to the nature of the dataset: on *Wiki10* and *delicious-200K*, our method is very close to the best results; on the other datasets, there is a gap between the performances of our approach and the others. The main difference between the two sets of corpora is the hierarchical latent organization of the labels: the performances are better with our approach when such hierarchical organization is deep like for *Wiki10* or *Delicious*. At the contrary, results with more flat hierarchies like *Amazon* did not succeed to reach competitive performances.

Effect of the embedding dimension Poincaré ball model is known for the ability to efficiently embed hierarchical element in a low dimensional space. We design an experiment to verify this hypothesis by training our model using different embedding sizes and by comparing it to the *AnnexML* algorithm without prepartitioning (corresponding in fact to an euclidean embedding). Table 4 shows the results for *wiki10*. For low dimensions (5 or 10), our approach slightly outperforms the euclidean embedding especially in the case of a single learner. With the growth of the number of dimensions, euclidean embedding tends to outperform our approach.

The dimension of the embedding space is crucial in the case of extreme classification, as it is a key factor for the inference time: the k-nearest neighbors algorithm used for the prediction is linear with respect to the embedding size, thus the complexity of the inference step can be drastically improved if a very low embedding size is sufficient for good prediction performances.

Ensemble Learning Ensemble learning in large scale multi-label classification is a key process to increase performances. As in the case of the state of the art algorithms, we found that using several learners (models) and aggregating their votes improve the performances. The results are reported in Table 5. Our approach seems to benefit from multiple models, but quickly reach a plateau around 6 learners. These experiments show that the presented approach is quite robust and the learned representations relatively stable compare to *annexML* and *SLEEC* model.

Analysis of the representation space In this section, we report qualitative results allowing to understand the properties of the inferred representation space. A first observation concerns the learned representation of the labels: as expected, the norm of the embedding representation of a label is linked to its frequency in the dataset: we observe that most common labels have small norm when the less common ones

[1] http://manikvarma.org/downloads/XC/XMLRepository.html

Dataset	Avg #examples per label	Avg #labels per example	#labels	#train
AmazonCat-13K	448.57	5.04	306,782	1,186,239
Wiki10-31K	8.52	18.64	30,938	14,146
Delicious-200K	72.29	75.54	205,443	196,606
WikiLSHTC-325K	17.46	3.19	325,056	1,778,351
Amazon-670K	4	5.45	670,091	490,449

Table 1: Main statistics of the dataset: the average number of examples per label, the average number of labels per example, the number of labels and the number of training examples.

Dataset	p@1/p@3/p@5
AmazonCat-13K	30.0%/18.8%/14.9%
Wiki10-31K	80.8%/50.5%/36.8%
Delicious-200K	38.7%/36.8%/35.5%
WikiLSHTC-325K	15.9%/6.03%/3.80%
Amazon-670K	$2.8e^{-3}$%/$2.7e^{-3}$%/$2.3e^{-3}$%

Table 2: Most-common baseline results, precision at 1/3/5 when the predicted labels are the most frequent labels of the training set.

have the largest norm. For instance, in the case of *wiki10*, Table 6 shows the norm and the frequency of some labels. This experiment indicates that our model is able to capture the latent hierarchical information among the labels. To confirm this fact, we learned a model in a simple two dimensions space. Figure 3.2 shows the label embeddings learned by this model. The color of the vector is depending on the frequency of the embedded label. The visualization shows that rare labels are very close to the boundary of the ball.

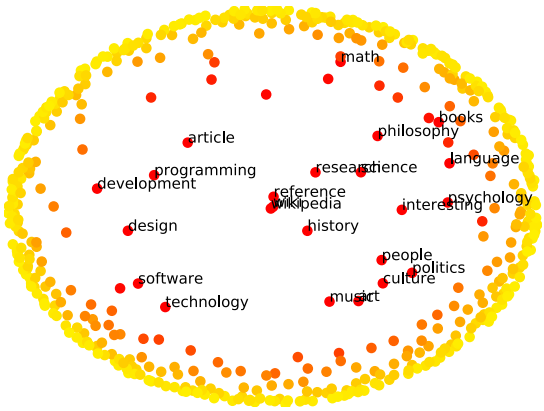


Figure 2: Label embeddings for *wiki10* in a two dimensions representation space. The color tends to red for more frequent labels and to yellow for rare labels.

4 Conclusion and Future Works

In this work we investigated the *Extreme Multi-Label classification* task using a hyperbolic representation learning paradigm. We propose an algorithm to learn a joint label/example representation. We investigate the strength of the hyperbolic space to represent a large amount of data and its capacity to represent and capture latent hierarchical information. Our procedure does not involve a preprocessing step when state of the art algorithms use a heavy preprocessing clustering procedure and/or pre-computation of example similarities. The conducted experiments have shown that for datasets with deep hierarchical label organization, our model reaches the state of the art algorithms and can outperform them in very low dimension. This is an important fact as the number of dimensions is a key factor for the inference time. However, on other datasets with a flat hierarchical structure, there is a real gap of performances. The first improvement under investigation concerns the loss function. The experiments show that the proposed loss function is able to represent well the labels, but lacks of local capacity to distinguish well examples with some labels in common. We intend to reinforce the proposed loss by adding a term able to bring closer examples with most similar labels and repulse examples sharing few labels, i.e. adding a term based on example similarity. A second improvement concerns the optimization procedure which has shown instability. A more adapted gradient descent can be derived specifically for the considered hyperbolic representation.

References

[BJK+15] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse Local Embeddings for Extreme Multi-label Classification. In *Advances in Neural Information Processing Systems 28*, pages 730–738. 2015.

Dataset		Hyp	SLEEC	Parabel	FastXML	PfastreXML	AnnexML
amazonCat	P@1	89.8%	90.5%	93.0%	93.1%	91.8%	93.6%
	P@3	73.5%	76.3%	79.16%	78.2%	78.0%	78.4%
	P@5	58.7%	61.5%	64.5%	63.4%	63.7%	63.3%
wiki10	P@1	85.6%	85.9%	84.3%	83.0%	83.6%	86.5%
	P@3	72.9%	73.0%	72.6%	67.5%	68.6%	74.3%
	P@5	62.4%	62.7%	63.4%	57.8%	59.1%	64.2%
Delicious-200K	P@1	46.5%	47.85%	47.0%	43.07%	41.72%	46.6%
	P@3	40.8%	42.21%	40.1%	38.66%	37.83%	40.8%
	P@5	37.8%	39.43%	36.6%	36.19%	35.58%	37.8%
WikiLSHTC-325K	P@1	43.2%	54.8%	65.0%	49.8%	56.0%	63.4%
	P@3	26.5%	33.4%	43.2%	33.1%	36.8%	40.7%
	P@5	19.4%	23.9%	32.1%	24.5%	27.1%	29.8%
Amazon-670K	P@1	31.1%	35.1%	44.9%	37.0%	39.5%	42.0%
	P@3	28.1%	31.3%	39.8%	33.3%	35.8%	36.7%
	P@5	26.0%	28.7%	36.0%	30.5%	33.1%	32.8%

Table 3: Results of our approach (Hyp) compared to several state of the art approaches. For *SLEEC* and *AnnexML*, the predictions are made by aggregating the label vote of 15 different models. *FastXML* and *PfastreXML* aggregate 50 trees prediction. *AnnexML* merging 1 to 3 classifiers results. All reported results come from the *Extreme Classification repository*, except for *AnnexML* for which the reported results came from [Tag17].

#Learners	# Dimensions	Hyperbolic	Euclidean
1	5	78.9/54.1/43.1	77.02/53.5/42.5
3	5	81.3/60.0/50.0	81.1/59.2/48.7
6	5	81.1/61.5/52.0	81.3/60.3/50.4
9	5	81.2/61.8/52.6	81.2/60.6/50.7
12	5	81.2/61.8/53.0	81.2/60.6/50.9
15	5	81.2/61.6/53.1	81.2/60.6/50.9
1	10	81.5/61.6/50.9	79.1/61.4, 50.4
3	10	82.8/66.5/56.0	83.0/66.2/55.0
6	10	82.6/67.4/57.5	83.2/67.3/56.9
9	10	82.6/67.6/57.7	83.1/67.5/57.4
12	10	82.5/67.4/57.7	83.1/67.4/57.6
15	10	82.6/67.6/58.0	83.1/67.5/57.5
1	25	82.9/67.7/57.1	82.5/67.8/57.5
3	25	84.8/70.1/59.7	84.8/71.0/60.5
6	25	84.6/71.0/60.3	85.6/71.7/61.5
9	25	84.5/71.0/60.5	85.5/77.9/61.7
12	25	84.5/71.1/60.7	85.8/71.9/61.7
15	25	84.6/71.3/60.7	85.8/72.0/67.9

Table 4: Influence of the size of the representations and the number of aggregated models for the *Wiki10-31K* corpus for the precision @1,3,5. Hyperbolic refers to our approach, euclidean refers to an euclidean embedding similar to the AnnexML algorithm without pre-clustering.

Dataset	# learners	P@1	P@3	P@5	SLEEC (15 Learners)
AmazonCat	1	86.7%	69.1%	54.5%	90.5%/76.3%/61.5%
	3	88.7%	72.1%	57.3%	
	6	89.4%	72.8%	58.2%	
	9	89.6%	73.3%	58.5%	
	12	89.7%	73.4%	58.6%	
	15	89.8%	73.5%	58.7%	
Wiki10-31k	1	83.9%	70.1%	59.7%	85.9%/73.0%/62.7%
	3	85.3%	72.3%	61.6%	
	6	85.7%	72.6%	62.1%	
	9	85.6%	72.8%	62.4%	
	12	85.5%	73.0%	62.5%	
	15	85.6%	72.9%	62.4%	
Delicious Large 200K	1	41.1%	35.4%	32.5%	47.9%/42.2%/39.4%
	3	44.6%	39.0%	36.1%	
	6	45.8%	40.2%	37.2%	
	9	46.3%	40.6%	37.6%	
	12	46.5%	40.8%	37.8%	
	15	46.5%	40.8%	37.8%	
WikiLSHTC-325K	1	36.4%	20.5%	14.5%	54.8%/33.4%/23.9%
	3	40.9%	24.1%	17.3%	
	6	42.4%	25.5%	18.5%	
	9	43.0%	26.2%	19.1%	
	12	43.2%	26.5%	19.4%	
Amazon-670K	1	19.3%	18.0%	17.1%	35.1%/31.3%/28.6%
	3	23.9%	21.8%	20.3%	
	6	27.4%	24.8%	23.04%	
	9	29.2%	26.4%	24.5%	
	12	30.3%	27.4%	25.4%	
	15	31.1%	28.1%	26.0%	

Table 5: Precision @k (1/3/5) depending on the number of aggregated models for our method and *SLEEC*.

Label Name	Norm	Frequency
wikipedia	0.6	80.7
wiki	6.8	41.9
reference	10.7	28.3
history	14.6	18.7
science	21.5	12.3
research	23.0	14.5

Table 6: Label sorted by the norm of their representation for the *wiki10* dataset.

9

[CL12] Yao-nan Chen and Hsuan-tien Lin. Feature-aware label space dimension reduction for multi-label classification. In *Advances in Neural Information Processing Systems 25*, pages 1529–1537, 2012.

[GBH18] Octavian Ganea, Gary Becigneul, and Thomas Hofmann. Hyperbolic Neural Networks. In *Advances in Neural Information Processing Systems 31*, pages 5345–5355. 2018.

[HKLZ09] Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang. Multi-Label Prediction via Compressed Sensing. In *Advances in Neural Information Processing Systems 22*, pages 772–780, 2009.

[JPV16] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications. In *Proceedings of the 22nd ACM SIGKDD*, pages 935–944, 2016.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[NK17] Maximillian Nickel and Douwe Kiela. Poincaré Embeddings for Learning Hierarchical Representations. In *Advances in Neural Information Processing Systems 30*, pages 6338–6347. 2017.

[PKH+18] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 993–1002, 2018.

[PV14] Yashoteja Prabhu and Manik Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 263–272, 2014.

[SDSGR18] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Re. Representation tradeoffs for hyperbolic embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4460–4469, 10–15 Jul 2018.

[Tag17] Yukihiro Tagami. Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD*, pages 455–464, 2017.

[TL12] Farbound Tai and Hsuan-Tien Lin. Multilabel classification with principal label space transformation. *Neural Comput.*, 24(9):2508–2542, September 2012.

[TLH17] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Enabling efficient question answer retrieval via hyperbolic neural networks. *CoRR*, abs/1707.07847, 2017.

[YHD+17] Ian E.H. Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. PPDsparse: A Parallel Primal-Dual Sparse Method for Extreme Classification. In *Proceedings of the 23rd ACM SIGKDD*, pages 545–553, 2017.

[YHZ+] Ian E H Yen, Xiangru Huang, Kai Zhong, Pradeep Ravikumar, and Inderjit S Dhillon. PD-Sparse : A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification. page 11.

[ZH05] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.

[ZLLG18] Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, pages 1–12, 2018.

Détection contextuelle d'anomalies à partir de séries temporelles multi-variées à base de modèle de Poisson

Seif-Eddine Benkabou[1], Khalid Benabdeslem[1], Vivien Kraus[1,2], Kilian Bourhis[2], et Bruno Canitia[2]

[1]Université Claude Bernard Lyon 1
[2]Lizeo Online Media Group

Résumé

Les séries temporelles multi-variées sont omniprésentes dans différents domaines, de la supervision des centres de données, aux données d'e-commerce en passant par les transactions financières. Ce type de données présente un important défi pour la détection d'anomalies à cause de la dépendance temporelle entre les observations. Dans ce papier, nous nous intéressons au problème de la détection non-supervisée des anomalies dans une série temporelle multi-variée, en se fondant d'une part sur la modélistion de l'aspect temporel de la série et d'autre part sur l'analyse des résidus de la phase de reconstruction. En fait, l'analyse des résidus a démontré sa performance dans des problèmes de détection d'anomalies classiques ; toutefois cette tâche est plus complexe pour des séries multi-variées puisque la dépendance temporelle entre les observations complique le processus de modélisation des résidus. Nous proposons une méthode d'apprentissage unifiée pour caractériser les résidus et leur cohérence avec l'aspect temporel de la série. Des expérimentations sur des jeux de données réels sont proposées et montrent l'efficacité de l'algorithme proposé.

Mots-clef : détection d'anomalies, séries temporelles, optimisation.

1 Introduction

La détection d'anomalie est une tâche importante dans plusieurs applications d'apprentissage automatique. Elle consiste à identifier la minorité des observations qui peuvent être considérées comme anormales par rapport à la majorité des données. En général, une anomalie est *une observation qui dévie tellement des autres que l'on pourrait penser qu'elle a été générée par un mécanisme différent* [Haw80]. Cette tâche a été appliquée dans plusieurs domaines comme la biologie, l'astronomie, l'économie, la détection d'intrusion et la finance [Agg13]. La détection d'anomalie a été étudiée pour différents types de données, y compris les données à grandes dimensions [PES01, EBES05, SC05], les graphes [CBK09, BSO$^+$09], ou encore les données temporelles [AS12, BSAT06, CMK08].

De plus, la détection d'anomalie pour les séries temporelles multi-variées peut être effectuée par une approche globale ou locale. La première signifie que les anomalies sont détectées directement parmi un ensemble de séries temporelles multi-variées [BBC18]. La seconde, qui est notre sujet principal dans ce papier, concerne la détection des anomalies au sein d'une série temporelle multi-variée [AY01] en prenant en compte l'aspect temporel de la série. Plusieurs approches ont été proposées pour détecter les observations anormales au sein d'une série temporelle multi-variée, même si la plupart des approches traditionnelles, comme l'estimation de la densité [YC02, BKNS00] et les techniques de classification [BSO$^+$09], peuvent être facilement adaptées au problème de détection d'anomalie locale. En effet, détecter les anomalies au sein d'une série multi-variée est plus difficile pour plusieurs raisons. D'une part, il est plus difficile de définir ce qu'est une anomalie pour ce type de données. Comme pour les séries temporelles uni-variées, certaines anomalies peuvent correspondre à des sous-séquences inhabituelles (par exemple la discordance) [KLF05] ou à des valeurs anormalement élevées dans une ou plusieurs séries temporelles. D'autre part, les anomalies peuvent correspondre à des changements anormaux des relations au sein d'un ensemble de variables, comme dans [CCJY08]. Dans ce papier, nous nous concentrons sur deux types d'anomalies qui peuvent survenir dans une série temporelle multi-variée : anomalies *globales*, et anomalies *locales* ou *contextuelles*.

Une anomalie globale est une observation qui diffère d'une certaine manière de la majorité des autres observations, indépendamment de sa position dans la série temporelle ; alors que l'anomalie contextuelle (ou locale) est une observation qui diffère des observations qui lui sont voisines dans le temps. Ainsi, une observation peut être anormale dans un contexte, et normale dans un autre, ce qui complique encore plus la tâche de détection. Alors qu'une anomalie globale peut être facilement détectée par des approches conventionnelles, la détection d'anomalies contextuelles offre plus de défis, car il est nécessaire de considérer les dépendances temporelles entre toutes les observations des séries temporelles. Ainsi, il paraît pertinent de proposer une méthodologie robuste pour détecter les observations anormales (locales et globales) dans un contexte général au sein d'une série temporelle multi-variée.

Détecter les données anormales via l'analyse des résidus a été initialement proposé par [SO11]. Cela consiste à étudier les résidus entre les données originales et les données estimées. D'autres méthodes sont fondées sur un processus de projection / reconstruction [GPST06], afin de faciliter la compréhension des anomalies. Certaines de ces approches sont fondées sur l'ACP, comme la méthode SPIRIT dans [PSF05], ou sur la projection comme le Delta Aléatoire dans [HT18]. En supposant que les anomalies sont généralement rares, les observations avec de grandes erreurs résiduelles auront plus de chances d'être anormales, car leur comportement n'est pas conforme aux observations de référence. Même si l'analyse des résidus constitue une bonne approche pour identifier les anomalies, cela reste difficile à adapter pour les données d'une série temporelle multi-variée, où il est insuffisant de considérer seulement les informations résiduelles des observations. Cela s'explique par leur grande dépendance les unes avec les autres. En effet, dans une série temporelle multi-variée, les observations ne sont ni indépendantes ni identiquement distribuées (*i.i.d.*), et malheureusement leurs dépendances temporelles complique la modélisation des résidus.

Dans ce papier, nous proposons une approche de détection d'anomalies générale via l'analyse des résidus. En réalité, nous montrons comment utiliser les résidus des observations issues des séries temporelles afin d'identifier et détecter les anomalies dans un cadre non-supervisé, sans connaissances a priori sur les anomalies. De plus, nous analysons la cohérence entre les résidus de la modélisation et l'aspect temporel des séries pour détecter les données anormales d'une manière unifiée. Les principales contributions de notre travail sont listées ci-après :

— la modélisation, de manière unifiée, des résidus des observations d'une série temporelle mutli-variée ;
— la proposition d'une approche probabiliste pour modéliser la dépendance temporelle au sein d'une série temporelle multi-variée en fonction des informations résiduelles ;
— la combinaison des résidus de la modélisation avec la dépendance temporelle au sein d'un même modèle pour la détection d'anomalies.

Le reste de ce papier est organisé comme suit. Dans la section 2, nous définissons le modèle que nous proposons pour la détection via l'analyse des résidus et la dépendance temporelle. Dans la section 3, nous présentons le problème d'optimisation de l'approche proposée et nous montrons comment le résoudre efficacement. De plus, nous discutons de l'analyse de la convergence et de la complexité temporelle de notre algorithme. Dans la section 4, nous apportons quelques résultats expérimentaux pour valider notre proposition. Finalement, dans la section 5, nous concluons notre travail et discutons de quelques perspectives de recherche.

2 Approche proposée pour la Détection d'Anomalies

Dans cette section, nous donnons d'abord quelques notations utilisées à travers ce papier et définissons formellement le problème de détection d'observations anormales dans de grandes séries temporelles multi-variées. Nous expliquons comment modéliser ces séries temporelles et la dépendance temporelle entre leurs observations, d'une part, et comment combiner cette dépendance temporelle avec l'information des résidus pour identifier à la fois les observations anormales locales et globales, d'autre part.

2.1 Définitions et notations

Dans cette section, nous présentons les notations utilisées dans ce papier. Soit $\mathbf{T}$ une série temporelle multi-variée de taille n. On note t_i les observations au sein $\mathbf{T}$. On rappelle que le but est de détecter les observations anormales dans une série temporelle multi-variée. Ainsi, on note que chaque observation t_i est vue comme un vecteur p-dimensionnel. Nous proposons donc de représenter cette série par une matrice n par p (par exemple, $\mathbf{T} \in \mathbb{R}^{n \times p}$) comme le montre la Figure 1.

Pour une telle matrice $\mathbf{T} = (t_{i,j}) \in \mathbb{R}^{n \times p}$, on note sa $i^{\text{ième}}$ ligne et sa $j^{\text{ième}}$ colonne respectivement t^i et t_j. Par la suite, on définit les différentes normes utilisées dans ce papier. La norme de Frobenius de $\mathbf{T} \in \mathbb{R}^{n \times p}$ est définie par : $\|\mathbf{T}\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{p} t_{ij}^2} =$

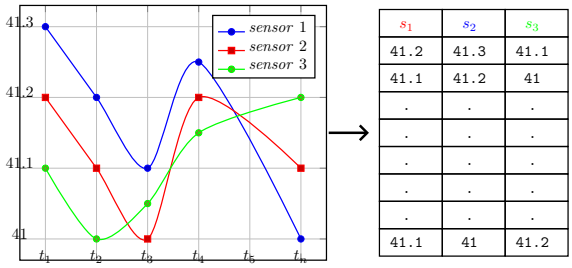


FIGURE 1 – Série temporelle multi-variée représenté par une matrice $(n \times p)$.

$\sqrt{\sum_{i=1}^{n} \|t^i\|_2^2}$. Nous utilisons les normes matricielles $\ell_{2,0}$ et $\ell_{2,1}$ qui sont respectivement définies par : $\|\mathbf{T}\|_{2,0} = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{p} |t_{ij}|^0} = \sum_{i=1}^{n} \|t^i\|_0$ et $\|\mathbf{T}\|_{2,1} = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{p} t_{ij}^2} = \sum_{i=1}^{n} \|t^i\|_2$.

2.2 Approximation non-supervisée

L'idée d'approximation non-supervisée a été introduite par les auteurs de [TL13]. Cela consiste à représenter chaque observation comme une combinaison linéaire de quelques autres observations importantes du jeu de données. Ainsi, pour donner une approximation d'une série temporelle multi-variée de manière non-supervisée, on commence par garder uniquement ses observations sans leurs dépendances temporelles. Soit $\hat{\mathbf{T}} \in \mathbb{R}^{n \times p}$ la matrice approchée de la série temporelle $\mathbf{T} \in \mathbb{R}^{n \times p}$; alors la matrice d'erreur d'approximation $\mathbf{T} - \hat{\mathbf{T}}$ peut être utilisée pour identifier les observations qui sont *globalement* anormales par rapport au jeu de données complet. Plusieurs approches peuvent être utilisées pour construire la matrice d'approximation $\hat{\mathbf{T}}$, et une façon simple de le faire est d'utiliser quelques observations représentatives [YBT06]. Pour une observation donnée, si elle peut être approchée par quelques observations pertinentes, la probabilité qu'elle soit anormale est faible. En revanche, si elle ne peut pas être bien approchée, elle n'est pas conforme au comportement normal du jeu de données. Dans notre cas, ce problème peut être formulé comme la minimisation d'une erreur quadratique entre $\mathbf{T}$ et sa matrice d'approximation $\hat{\mathbf{T}} = \mathbf{W}^\top \mathbf{T}$:

$$\min_{\mathbf{W}} \|\mathbf{T} - \mathbf{W}^\top \mathbf{T}\|_F^2 + \alpha \|\mathbf{W}\|_{2,0} \tag{1}$$

Où $\mathbf{W} \in \mathbb{R}^{n \times n}$ est une matrice de coefficients telle que chaque observation t^i de $\mathbf{T}$ peut être reconstruite comme une combinaison linéaire d'autres observations, et le vecteur colonne w_i de la matrice $\mathbf{W}$ contient les coefficients d'une telle combinaison. Les coefficients reflètent la contribution de chaque observation au processus de reconstruction de t^i. Par exemple, si $\mathbf{W}_{i,j} = 0$, alors l'observation t^j ne contribue pas à la reconstruction de t^i. L'observation reconstruite $\hat{t^i}$ de t^i est obtenue par $w_i \mathbf{T}$. En ce qui concerne les observations représentatives, on suppose qu'il y en a peu, afin d'avoir une matrice de coefficients $\mathbf{W}$ creuse. Pour ce faire, la contrainte de parcimonie $\|\mathbf{W}\|_{2,0}$ sur les lignes de $\mathbf{W}$ est utilisée pour restreindre le nombre d'observation représentatives. Ainsi, on utilise un paramètre de régularisation α pour contrôler le nombre d'observations à considérer comme représentatives dans $\mathbf{T}$. L'idée est d'avoir une matrice $\mathbf{W}$ creuse dans laquelle tous les vecteurs lignes w^i sont nuls, sauf ceux qui sont représentatifs de la série temporelle. Cependant, le problème de minimisation (1) est difficile à cause de l'utilisation de la norme $\ell_{2,0}$ Pour pallier ce problème, ils ont proposé de relaxer ce problème en utilisant la norme $\ell_{2,1}$ de $\mathbf{W}$, qui est l'enveloppe convexe minimale de la norme $\ell_{2,0}$. Par ailleurs, on minimise $\|\mathbf{W}\|_{2,1}$ pour obtenir approximativement les mêmes résultats qu'avec $\|\mathbf{W}\|_{2,0}$. Ainsi, le problème de minimisation (1) peut être réécrit :

$$\min_{\mathbf{W}} \|\mathbf{T} - \mathbf{W}^\top \mathbf{T}\|_F^2 + \alpha \|\mathbf{W}\|_{2,1} \tag{2}$$

La détection d'observations anormales à partir de l'analyse des résidus a été proposée initialement par [SO11] pour des problèmes de régression pénalisée. Dans notre cas, on utilise la matrice des résidus $\mathbf{R} = \mathbf{T} - \mathbf{W}^\top \mathbf{T} - \Theta$ où Θ représente la matrice d'erreur aléatoire, par hypothèse distribuée normalement selon [SO11], pour détecter les points anormaux. En premier lieu, on suppose que ces observations anormales sont assez différentes et dévient significativement de l'ensemble des observations de la série temporelle $\mathbf{T}$. Selon cette hypothèse, les observations anormales ne devraient pas être bien représentées à l'issue de l'étape de reconstruction. Ceci impliquerait une grande erreur de reconstruction, et pourrait être analysé avec la matrice des résidus $\mathbf{R}$. Ainsi, une grande norme ℓ_2 pour la ligne r^i de $\mathbf{R}$ indiquerait une déviation significative de la majorité des observations. Pour cette raison, on intègre la matrice des résidus $\mathbf{R}$ dans notre fonction objective et le problème d'optimisation devient :

$$\min_{\mathbf{W},\mathbf{R}} \|\mathbf{T} - \mathbf{W}^\top \mathbf{T} - \mathbf{R}\|_F^2 + \alpha \|\mathbf{W}\|_{2,1} + \beta \|\mathbf{R}\|_{2,1} \tag{3}$$

Sous l'hypothèse que les anomalies sont rares, nous proposons d'ajouter une contrainte de parcimonie sur la matrice $\mathbf{R}$, dans (3) avec la norme $\ell_{2,1}$, de sorte que la plupart des lignes soient nulles et seulement certaines (correspondant aux anomalies) soient non-nulles.

2.3 Modélisation temporelle

Comme on peut le voir, le problème de minimisation ci-dessus (3) nous permet de détecter des observations anormales en exploitant la matrice des résidus issue de l'étape d'approximation. Cependant, l'aspect temporel et séquentiel des observations de la série $\mathbf{T}$ ne sont pas pris en compte. En nous limitant à l'information des résidus uniquement, notre approche est incapable de détecter les anomalies contextuelles. Ainsi, pour détecter ce genre d'anomalies, on se fonde sur l'hypothèse suivante à propos de l'étape d'approximation et l'information des résidus : *les résidus d'une observation normale doivent être assez similaires aux résidus des observations voisines dans le temps.* Ainsi, si l'observation t^i est temporellement proche de l'observation t^j, alors la différence entre leurs résidus $\|r^i - r^j\|_2$ doit être proche de zéro ou égale à zéro. En généralisant cette notion à la dépendance temporelle pour toutes les observations, la somme des différences entre les résidus des observations proches doit être minimale :

$$\frac{1}{2}\sum_{i,j}\left(\|r^i - r^j\|_2^2 \times \delta_{i,j}\right) \tag{4}$$

Où $\delta_{i,j}$ est une fonction qui modélise la dépendance temporelle entre les observations t^i et t^j. Cette fonction retourne une valeur réelle positive qui mesure la proximité temporelle entre deux observations (Figure 2). Plus cette valeur est élevée, plus les observations t^i et t^j sont proches dans le temps. Quand les observations sont assez éloignées sur l'axe temporel, la fonction δ tend vers zéro, ce qui implique que la différence entre leurs résidus ne contribue pas à la minimisation de (4). En suivant notre hypothèse, pour toute observation t^i, les valeurs retournées par $\delta_{i,.}$ doivent être décroissantes, et plus on s'éloigne de cette observation sur l'axe temporel, plus les valeurs doivent être proches de zéro. Nous proposons de représenter la fonction $\delta_{.,.}$ par les valeurs de probabilité de la distribution de Poisson pour le cas $\lambda = 1$:

$$\mathbb{P}(k) = \mathbb{P}(X = k) = \frac{\lambda^k}{k!}\,\mathrm{e}^{-\lambda} \tag{5}$$

$$\delta_{i,j} = \mathbb{P}(k = |i-j|) = \frac{e^{-1}}{|i-j|!} \tag{6}$$

Ainsi, plus on s'éloigne de la position i, dans les deux sens de l'axe temporel, plus la probabilité d'être proche de cette position décroît rapidement. L'équation (4) peut être réécrite en (8), où $\mathbf{M} \in \mathbb{R}^{n\times n}$ est une matrice symétrique qui modélise la dépendance temporelle entre toutes les observations d'une série multi-variée $\mathbf{T}$, en utilisant la matrice des résidus $\mathbf{R}$, telle que :

$$m_{ij} = \begin{cases} \delta_{i,j} = \frac{e^{-1}}{|i-j|!}, & \text{si } i \neq j \\ 0, & \text{si } i = j \end{cases} \tag{7}$$

$\mathbf{D}$ est une matrice diagonale telle que $d_{ii} = \sum_{j=1}^{n} m_{ij}$ et $\mathbf{L} = (\mathbf{D} - \mathbf{M})$ est la matrice Laplacienne.

$$\begin{aligned}
&\frac{1}{2}\sum_{i,j}\|r^i - r^j\|_2^2 \frac{e^{-1}}{|i-j|!} \\
&= \frac{1}{2}\sum_{i,j}\frac{\|r^i\|_2^2 e^{-1}}{|i-j|!} + \frac{\|r^j\|_2^2 e^{-1}}{|i-j|!} - \sum_{i,j}\frac{{r^i}^\top r^j e^{-1}}{|i-j|!} \\
&= \frac{1}{2}\sum_{i}\|r^i\|_2^2 d_{ii} + \frac{1}{2}\sum_{i}\|r^i\|_2^2 d_{ii} - \sum_{i,j}\frac{{r^i}^\top r^j e^{-1}}{|i-j|!} \\
&= tr(\mathbf{R}^\top\mathbf{D}\mathbf{R} - \mathbf{R}^\top\mathbf{M}\mathbf{R}) = tr(\mathbf{R}^\top(\mathbf{D}-\mathbf{M})\mathbf{R}) \\
&= tr(\mathbf{R}^\top\mathbf{L}\mathbf{R})
\end{aligned} \tag{8}$$

2.4 Détection locale : LADOP

Dans la fonction objective de (3), les séries temporelles $\mathbf{T}$ sont vues comme des matrices n par d sans relations entre leurs lignes. En effet, la fonction est fondée sur une forte hypothèse, à savoir que les observations t^i de $\mathbf{T}$ sont indépendantes et identiquement distribuées. Cependant, cela n'est pas le cas dans les séries temporelles multi-variées où les observations sont temporellement dépendantes les unes des autres dans un certain contexte. Ainsi, nous proposons d'intégrer le terme de dépendance temporelle probabiliste dans (8), en se fondant sur (3) pour ajouter l'aspect temporel des observations dans le processus de modélisation des résidus. Ensuite, on obtient la formulation finale en fonction de $\mathbf{W}$ et $\mathbf{R}$, qui permet de détecter les anomalies contextuelles et globales au sein des séries temporelles multi-variées :

$$\begin{aligned}
\Phi(\mathbf{W},\mathbf{R}) =& \|\mathbf{T} - \mathbf{W}^\top\mathbf{T} - \mathbf{R}\|_F^2 + \alpha\|\mathbf{W}\|_{2,1} + \beta\|\mathbf{R}\|_{2,1} \\
&+ \gamma tr(\mathbf{R}^\top\mathbf{L}\mathbf{R})
\end{aligned} \tag{9}$$

Où γ est un paramètre de régularisation visant à établir un compromis entre la modélisation des résidus et la dépendance temporelle. En apprenant et en exploitant la matrice des résidus $\mathbf{R}$, on peut calculer le rang des anomalies selon leurs résidus.

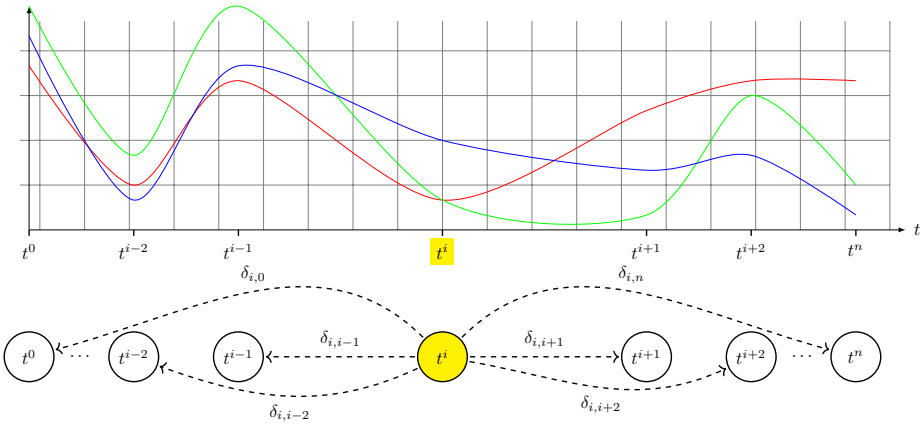


FIGURE 2 – Dépendance temporelle probabiliste d'une série temporelle multi-variée.

3 Problème d'optimisation

Dans cette section, nous commençons par détailler le modèle d'optimisation que nous avons utilisé pour résoudre le problème de minimisation décrit par l'équation (9). Ensuite, nous présentons l'algorithme issue de la résolution, accompagné d'une analyse de sa convergence et de sa complexité. La fonction objective présentée dans l'équation (9) n'est pas simultanément convexe par rapport à $\mathbf{W}$ et $\mathbf{R}$. De plus, la pénalisation de la norme $\ell_{2,1}$ rend cette fonction non-lisse. Ceci nous pousse à utiliser une optimisation alternée de $\mathbf{W}$ et $\mathbf{R}$ pour minimiser la fonction. Ce type d'optimisation consiste à fixer une variable et optimiser par rapport à l'autre. Par exemple, en fixant $\mathbf{W}$, la fonction objective devient convexe par rapport à $\mathbf{R}$ et vice versa.

Optimisation par rapport à R : Tout d'abord, fixons $\mathbf{W}$ et supprimons tous les termes qui ne dépendent pas de $\mathbf{R}$. Ainsi, le problème de minimisation (9) devient :

$$\min_{\mathbf{R}} \Phi(\mathbf{R}) = \|\mathbf{T}-\mathbf{W}^\top\mathbf{T}-\mathbf{R}\|_F^2+\beta\|\mathbf{R}\|_{2,1}+\lambda tr(\mathbf{R}^\top\mathbf{L}\mathbf{R}) \tag{10}$$

En calculant la dérivée de $\Phi(\mathbf{R})$ par rapport à $\mathbf{R}$, on obtient :

$$\frac{\partial\Phi(\mathbf{R})}{\partial\mathbf{R}} = \mathbf{W}^\top\mathbf{T} - \mathbf{T} + \mathbf{R} + \beta\mathbf{D}_R\mathbf{R} + \lambda\mathbf{L}\mathbf{R} = 0 \tag{11}$$

Où $\mathbf{D}_R$ est une matrice diagonale telle que ses éléments $d_{Rii} = \frac{1}{2\|r^i\|_2+\epsilon}$. Les matrices $\mathbf{I}$ et $\beta\mathbf{D}_R$ sont diagonales à valeurs positives, elles sont donc semi-définies positives. La matrice Laplacienne $\mathbf{L}$ est aussi semi-définie positive. Ainsi, la somme de ces trois matrices, $\mathbf{I}+\beta\ D_R+\lambda\mathbf{L}$, est également semi-définie positive, ce qui implique que $\mathbf{R}$ a une solution analytique à

$$\mathbf{R} = (\mathbf{I} + \beta\mathbf{D}_R + \lambda\mathbf{L})^{-1}(\mathbf{T} - \mathbf{W}^\top\mathbf{T}). \tag{12}$$

Optimisation par rapport à W : De la même façon, pour $\mathbf{W}$, on fixe $\mathbf{R}$ et on enlève les termes qui ne dépendent pas de $\mathbf{W}$. Le problème d'optimisation devient :

$$\min_{\mathbf{W}} \Phi(\mathbf{W}) = \|\mathbf{T} - \mathbf{W}^\top\mathbf{T} - \mathbf{R}\|_F^2 + \alpha\|\mathbf{W}\|_{2,1} \tag{13}$$

De la même manière, en annulant la dérivée de $\Phi(\mathbf{W})$ par rapport à $\mathbf{W}$, on obtient :

$$\frac{\partial\Phi(\mathbf{W})}{\partial\mathbf{W}} = (\mathbf{T}\mathbf{T}^\top+\alpha\mathbf{D}_W)\mathbf{W}-\mathbf{T}\mathbf{T}^\top+\mathbf{T}\mathbf{R}^\top = 0 \tag{14}$$

Où $\mathbf{D}_W$ est une matrice diagonale telle que ses éléments $d_{Wii} = \frac{1}{2\|w^i\|_2+\epsilon}$. Les matrices $\mathbf{T}\mathbf{T}^\top$ et $\alpha\mathbf{D}_W$ sont semi-définies positives, et leur somme $\alpha\mathbf{D}_W+\mathbf{T}\mathbf{T}^\top$ est aussi semi-définie positive, ce qui implique que $\mathbf{W}$ a une solution analytique à (13) :

$$\mathbf{W} = (\mathbf{T}\mathbf{T}^\top + \alpha\mathbf{D}_W)^{-1}(\mathbf{T}\mathbf{T}^\top - \mathbf{T}\mathbf{R}^\top) \tag{15}$$

Nous pouvons donc résumer tous les développements mathématiques ci-dessus dans l'Algorithme 1 que nous appelons **LADOP** (pour ***L**ocal **A**nomaly **D**etection based **O**n **P**oisson model*).

Dans cet algorithme, on commence par initialiser les deux matrices $\mathbf{D}_R$ et $\mathbf{D}_W$ par la matrice identité, et la matrice des résidus $\mathbf{R}$ par $(\mathbf{I} + \beta\mathbf{D}_R + \lambda\mathbf{L})^{-1}\mathbf{T}$, une

Algorithm 1 LADOP

Entrée : Série temporelle multi-variée sous forme d'une matrice $\mathbf{T}$ de dimension $n \times p$. Paramètres de régularisation α, β et γ.
Sortie : Score d'anomalie de tous les points de la série temporelle $\mathbf{T}$.
Initialisation : $\mathbf{M}(i,j) \leftarrow \frac{e^{-1}}{|i-j|!}$, $\mathbf{D}(i,i) \leftarrow \sum_{j=1}^{n} m_{ij}$, $\mathbf{L} \leftarrow \mathbf{D} - \mathbf{M}$
$h \leftarrow 0$, $\mathbf{D}_{R_h} \leftarrow I$ et $\mathbf{D}_{W_h} \leftarrow I$
$\mathbf{R}_h \leftarrow (\mathbf{I} + \beta \mathbf{D}_R + \gamma \mathbf{L})^{-1} T$
while le critère de convergence n'est pas respecté **do**
  $\mathbf{W}_{h+1} \leftarrow (\mathbf{T}\mathbf{T}^\top + \alpha \mathbf{D}_{W_h})^{-1}(\mathbf{T}\mathbf{T}^\top - \mathbf{T}\mathbf{R}_h^\top)$
  $\mathbf{D}_{W_{h+1}}(i,i) \leftarrow \frac{1}{2\|w_{h+1}^i\|_2 + \epsilon}$
  $\mathbf{R}_{h+1} \leftarrow (\mathbf{I} + \beta \mathbf{D}_{R_h} + \lambda \mathbf{L})^{-1}(\mathbf{T} - \mathbf{W}_{h+1}^\top \mathbf{T})$
  $\mathbf{D}_{R_{h+1}}(i,i) \leftarrow \frac{1}{2\|r_{h+1}^i\|_2 + \epsilon}$
  $h \leftarrow h + 1$
end while
Calcul des scores d'anomalie de tous les t_i dans $\mathbf{T}$ comme $\|r_h^i\|_2$

approximation de (12). Ensuite, on effectue l'optimisation alternée. Ainsi, on fixe $\mathbf{R}$ pour mettre à jour $\mathbf{W}$ et vice versa, en un processus itératif jusqu'à la convergence de la fonction objective dans (9). Notons que $\|w^i\|_2$ et $\|r^i\|_2$ pourraient être proches de zéro sans nécessairement l'atteindre concrètement. Cependant, ils pourraient être théoriquement nuls (à cause de la contrainte de parcimonie), auquel cas on ajoute un petit epsilon à leurs définitions pour éviter de diviser par zéro lors de l'inversion des matrices diagonales $\mathbf{D}_W$ et $\mathbf{D}_R$ pour le calcul respectif des solution de $\mathbf{W}$ et $\mathbf{R}$. Une fois la convergence atteinte, le score d'anomalie pour chaque observation t^i est calculé selon la norme ℓ_2 correspondante dans la matrice des résidus $\mathbf{R}$ (par exemple, $\|r^i\|_2$). Ainsi, les observations avec une grande norme sont des anomalies plus probables. Enfin, on trie les lignes t^i par ordre décroissant de leur score.

3.1 Analyse de convergence

Dans cette partie, nous montrons que l'optimisation alternée proposée dans l'Algorithme 1, qui résout une version relaxée du problème (9), permet une décroissance monotone de la fonction objective dans (9). Pour prouver cette propriété de notre algorithme, nous suivons l'approche proposée par les auteurs dans [LDHL17], qui consiste à utiliser le lemme suivant (de [NHCD10]) :

Lemme 3.1. *Pour tout vecteur non nul $x, y \in \mathbb{R}^p$, l'inégalité suivante est vérifiée :*

$$\|x\|_2 - \frac{\|x\|_2^2}{2\|y\|_2} \leq \|y\|_2 - \frac{\|y\|_2^2}{2\|y\|_2} \tag{16}$$

Théorème 3.2. *Le processus d'optimisation alternée présenté dans l'Algorithme 1 mène à une décroissance monotone de la valeur de la fonction objective* (9).

Démonstration. À l'itération h, on fixe $\mathbf{R}$ pour obtenir $\mathbf{W}_{h+1}$ selon (15). En obtenant W_{h+1}, on peut calculer $\mathbf{R}_{h+1}$ via (12). Ensuite, on alterne entre ces deux calculs jusqu'à la convergence. Ainsi, pour montrer que la minimisation de la fonction objective Φ par rapport à $\mathbf{W}$ et $\mathbf{R}$ dans (9) converge, il suffit de montrer :

$$\Phi(\mathbf{W}_{h+1}, \mathbf{R}_{h+1}) \leq \Phi(\mathbf{W}_{h+1}, \mathbf{R}_h) \leq \Phi(\mathbf{W}_h, \mathbf{R}_h) \tag{17}$$

$\Phi(\mathbf{W}_{h+1}, \mathbf{R}_h) \leq \Phi(\mathbf{W}_h, \mathbf{R}_h)$ **:** Après avoir défini $\mathbf{R}_h$, on obtient $\mathbf{W}_{h+1}$ par (15), ce qui implique que $\mathbf{W}_{h+1}$ minimise Φ défini par (13) :

$$\mathbf{W}_{h+1} = \underset{\mathbf{W}}{\operatorname{argmin}} \|\mathbf{T} - \mathbf{W}^\top \mathbf{T} - \mathbf{R}_h\|_F^2 + \alpha \|\mathbf{W}\|_{2,1} \tag{18}$$

Étant donné que la norme $\ell_{2,1}$ de W n'est en réalité pas lisse dans (18), on la relaxe en utilisant la trace des deux matrices $(\mathbf{W}_{h+1}\mathbf{D}_W\mathbf{W}_{h+1})$ et $(\mathbf{W}_h\mathbf{D}_W\mathbf{W}_h)$. Ainsi, comme $\mathbf{W}_{h+1}$ minimise dans l'algorithme la version relaxée, on sait (19) :

$$\begin{aligned} &\|\mathbf{T} - \mathbf{W}_{h+1}^\top \mathbf{T} - \mathbf{R}_h\|_F^2 + \alpha tr(\mathbf{W}_{h+1}^\mathrm{T} \mathbf{D}_W \mathbf{W}_{h+1}) \\ &\leq \|\mathbf{T} - \mathbf{W}_h^\top \mathbf{T} - \mathbf{R}_h\|_F^2 + \alpha tr(\mathbf{W}_h^\mathrm{T} \mathbf{D}_W \mathbf{W}_h) \end{aligned} \tag{19}$$

D'un autre côté, selon le lemme (3.1), pour chaque vecteur ligne on a :

$$\begin{gathered} \|w_{h+1}^i\|_2 - \frac{\|w_{h+1}^i\|_2^2}{2\|w_h^i\|_2} \leq \|w_h^i\|_2 - \frac{\|w_h^i\|_2^2}{2\|w_h^i\|_2} \\ \text{donc} \\ \sum_{i=1}^{n} \left(\|w_{h+1}^i\|_2 - \frac{\|w_{h+1}^i\|_2^2}{2\|w_h^i\|_2} \right) \\ \leq \sum_{i=1}^{n} \left(\|w_h^i\|_2 - \frac{\|w_h^i\|_2^2}{2\|w_h^i\|_2} \right) \\ \text{donc} \\ \left(\|\mathbf{W}_{h+1}\|_{2,1} - \sum_{i=1}^{n} \frac{\|w_{h+1}^i\|_2^2}{2\|w_h^i\|_2} \right) \\ \leq \left(\|\mathbf{W}_h\|_{2,1} - \sum_{i=1}^{n} \frac{\|w_h^i\|_2^2}{2\|w_h^i\|_2} \right) \end{gathered} \tag{20}$$

De (19) on déduit l'inégalité suivante :

$$\begin{aligned}&\|\mathbf{T}-\mathbf{W}_{h+1}^{\top}\mathbf{T}-\mathbf{R}_h\|_F^2+\alpha\|\mathbf{W}_{h+1}\|_{2,1}\\&-\alpha\left(\|\mathbf{W}_{h+1}\|_{2,1}-\sum_{i=1}^{n}\frac{\|w^i{}_{h+1}\|_2^2}{2\|w^i{}_h\|_2}\right)\\&\leq\|\mathbf{T}-\mathbf{W}_h^{\top}\mathbf{T}-\mathbf{R}_h\|_F^2+\alpha\|\mathbf{W}_h\|_{2,1}.\\&-\alpha\left(\|\mathbf{W}_h\|_{2,1}-\sum_{i=1}^{n}\frac{\|w^i{}_h\|_2^2}{2\|w^i{}_h\|_2}\right)\end{aligned}\tag{21}$$

Enfin, en intégrant les inégalités (20) dans (21), on obtient :

$$\begin{aligned}&\|\mathbf{T}-\mathbf{W}_{h+1}^{\top}\mathbf{T}-\mathbf{R}_h\|_F^2+\alpha\|\mathbf{W}_{h+1}\|_{2,1}\\&\leq\|\mathbf{T}-\mathbf{W}_h^{\top}\mathbf{T}-\mathbf{R}_h\|_F^2+\alpha\|\mathbf{W}_h\|_{2,1}.\end{aligned}\tag{22}$$

Ce qui implique que :

$$\Phi(\mathbf{W}_{h+1},\mathbf{R}_h)\leq\Phi(\mathbf{W}_h,\mathbf{R}_h)\tag{23}$$

$\Phi(\mathbf{W}_{h+1},\mathbf{R}_{h+1})\leq\Phi(\mathbf{W}_{h+1},\mathbf{R}_h)$ **:** se montre en suivant les mêmes étapes que pour $\mathbf{W}$.

Le fait que les deux inégalités ci-dessus soient valides, prouve que le processus d'optimisation alternée présenté dans l'Algorithme 1 conduit à une décroissance monotone de la valeur de la fonction objective dans (9). □

3.2 Analyse de la complexité

Dans cette section, nous analysons la complexité temporelle de l'algorithme proposé. En effet, mettre à jour les deux matrices $\mathbf{W}$ et $\mathbf{R}$ représente les deux opérations les plus coûteuses de l'algorithme à cause des inversions des deux matrices, ou plutôt la résolution de deux systèmes linéaires. Pour éviter ces opérations matricielles coûteuses, on peut utiliser deux observations.

En premier lieu, la mise à jour de $\mathbf{W}$ peut être réécrite en utilisant l'inégalité matricielle de *Woodbury* (24) [Hig02], pour transformer la résolution du système linéaire $(n\times n)$ en une résolution $(p\times p)$; quitte à inverser $\mathbf{D}_W$ qui est une matrice diagonale. Ainsi, la complexité totale de la mise à jour de $\mathbf{W}$ est $\mathcal{O}(n^2p)$, à cause du calcul du terme $\mathbf{T}\mathbf{T}^{\top}-\mathbf{T}\mathbf{R}_h^{\top}$.

$$\begin{aligned}&(\mathbf{T}\mathbf{T}^{\top}+\alpha\mathbf{D}_W)^{-1}=\\&\frac{1}{\alpha}\mathbf{D}_W^{-1}-\frac{1}{\alpha^2}\mathbf{D}_W^{-1}\mathbf{T}(\mathbf{I}_p+\frac{1}{\alpha}\mathbf{T}^{\top}\mathbf{D}_W^{-1}\mathbf{T})^{-1}\mathbf{T}^{\top}\mathbf{D}_W^{-1}\end{aligned}\tag{24}$$

De plus, les valeurs de la matrice Laplacienne $\mathbf{L}$ tendent vers 0 en s'éloignant de la diagonale. Donc, sous l'hypothèse qu'il existe un entier ω tel que $\omega!\gg 1$, on peut approximer $\mathbf{L}$ par une matrice-bande creuse de largeur de bande ω. Ainsi, la mise à jour de $\mathbf{R}$ ne nécessite qu'une résolution d'un système linéaire creux par bande, ce qui donne une complexité temporelle de seulement $\mathcal{O}(\omega^2n)$ [KS13]. Une valeur constante pour ω donne une complexité temporelle globale de $\mathcal{O}(n^2p)$ à cause du produit $\mathbf{W}_{h+1}^{\top}\mathbf{T}$.

4 Expérimentations

Dans cette section, nous présentons nos expérimentations pour vérifier l'efficacité et la performance de notre modèle part rapport à d'autres approches pertinentes de la détection d'anomalies. Tout d'abord, nous présentons les jeux de données utilisés, puis nous décrivons le paramétrage des expérimentations et enfin les résultats obtenus.

4.1 Données et méthodes

Pour comparer différents algorithmes de détection d'anomalies, nous avons besoin de jeux de données dédiés à cette tâche. Cependant, il n'existe pas, à notre connaissance, de jeux de données publics où les observations anormales au sein d'une série multivariée sont connues à l'avance. Pour surmonter cette difficulté, nous avons mené des expériences sur six jeux de données publics, disponibles depuis le projet SPIRIT [1] et le dépôt UCI [2]. Ensuite, nous les avons modifiés par l'Algorithme 2, que nous avons développé afin d'injecter des anomalies contextuelles. Cet algorithme nous permet d'obtenir des observations labellisées au sein de chaque série temporelle (observation Normale ou Anormale). Concrètement, nous injectons N anomalies dans la série temporelle multi-variée $\mathbf{T}$ en échangeant aléatoirement les lignes de sa matrice, puis en enregistrant les indices ainsi échangés dans un ensemble "indSamples" pré-initialisé. Afin d'assurer que les observations échangées soient suffisamment différentes les unes des autres, nous avons ajouté une contrainte. C'est elle qui va assurer que pour chaque échange, au moins q séries temporelles suffisamment différentes soient échangées. Cette différence doit être plus grande qu'un facteur (σ) fois la déviation standard de $\mathbf{T}$. Pour nos expérimentations, nous avons fixé $N=1\%\times n$, $q=40\%\times p$ et $\sigma=2.5$ sauf pour le jeu de données "Q8calibVolt" $(\sigma=1.5)$.

1. http://www.cs.cmu.edu/afs/cs/project/spirit-1/www/
2. https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+

Algorithm 2 Injection d'Anomalies contextuelles

Entrées : $\mathbf{T} \in \mathbb{R}^{n \times p}$, N, q,σ
Sorties : $\mathbf{T}$ modifiée, indSamples
Initialisation : indSamples $\leftarrow \varnothing$
while taille de indSamples $< N$ **do**
  Tirer aléatoirement $(i, j) \in \{1, \ldots, n\}^2$ tel que ni i ni j n'ont été sélectionnés dans indSamples
  if $\text{Card}\{k; |\mathbf{T}_{ik} - \mathbf{T}_{jk}| > \sigma \times \text{std}(\mathbf{T}_{\cdot k})\} \geq q$ **then**
    Mettre à jour $\mathbf{T}$: Échanger les lignes i et j
    Ajouter (i, j) à indSamples
  end if
end while

Les caractéristiques de toutes les séries temporelles utilisées sont listées dans la Table 1. Nous avons comparé le LADOP avec quatre approches effectuant la détection d'anomalies. Deux d'entre elles utilisent l'estimation de densité : **Local Outlier Factor (LOF)** [BKNS00] et **Parzen-Window** [YC02] tandis que les autres utilisent la projection et la reconstruction : **SPIRIT** [PSF05] et **Random Projection (RP)** [HT18].

4.2 Résultats

Bien que la détection d'anomalies non-supervisée n'utilise pas concrètement l'information des labels, ils sont tout de même nécessaires pour l'évaluation et la comparaison, puisque la comparaison de la performance d'approches non-supervisées pour la détection d'anomalies n'est pas aussi facile que dans le cadre supervisé. En revanche, lorsque l'on veut simplement comparer le score d'exactitude (accuracy) du modèle ou sa précision et son rappel, l'ordre des anomalies doit être pris en compte. En classification, une observation mal classifiée est pour sûr une erreur. Mais cela est différent pour la détection d'anomalies dans un cadre non-supervisé. Par exemple, si une grande série temporelle multi-variée contient dix anomalies classées parmi le top-15 des anomalies, alors c'est un bon résultat, même si ce n'est pas parfait. Par conséquent, une stratégie d'évaluation commune pour les approches non-supervisées de détection d'anomalies est de classer les résultats selon le score d'anomalie et d'appliquer itérativement un seuil sur le classement ainsi obtenu. On obtient ainsi n couples de valeurs (taux de vrai positif et taux de faux positif), qui forment nos fonctions d'efficacité du récepteur (courbe ROC). Ensuite, l'aire sous la courbe (AUC), l'intégrale de la courbe ROC, peut être utilisée comme une mesure de performance de la détection d'anomalies. De ce fait, nous utilisons l'AUC comme mesure d'évaluation dans nos expérimentations. Les trois différents paramètres de notre modèle sont sélectionnés comme puissances de 10, avec un exposant choisi uniformément dans l'intervalle $[-3, +3]$.

Par la suite, nous comparons la performance du LADOP avec les méthodes de détection d'anomalies précédemment citées. Cette comparaison concerne les performances selon la courbe ROC et l'AUC, que nous présentons respectivement dans la Figure 3 et la Table 1. On peut observer que notre modèle obtient toujours la meilleure performance pour la détection d'anomalies. Grâce à sa modélisation efficace, la dépendance temporelle fondée sur la distribution de Poisson permet à notre algorithme de détecter les anomalies globales et contextuelles au cours du temps. De plus, de par son paradigme non-supervisé, notre algorithme tire parti de l'analyse des résidus pour la détection, sans aucune connaissance au préalable. En effet, dans les cas réels, les propriétés des anomalies d'une série temporelle sont souvent inconnues à l'avance.

5 Conclusion

Dans ce papier, nous proposons un modèle général pour la détection d'anomalies dans une série temporelle multi-variée. Pour cela, nous combinons l'analyse résiduelle, via une approximation non-supervisée, avec un modèle de dépendance temporelle probabiliste d'une série multi-variée, afin de détecter les deux types d'observations anormales (globales et locales) d'une manière unifiée. Les expériences sur les jeux de données comparatifs montrent que notre proposition obtient le meilleur score d'AUC comparée aux autres méthodes de détection d'anomalies. Malgré le fait que la dépendance probabiliste que nous proposons pour modéliser l'aspect temporel des séries donne de bons résultats, nous pensons que la stratégie d'apprentissage de cette dépendance pourrait être encore plus efficace, ce qui relaxerait légèrement l'hypothèse initiale.

Références

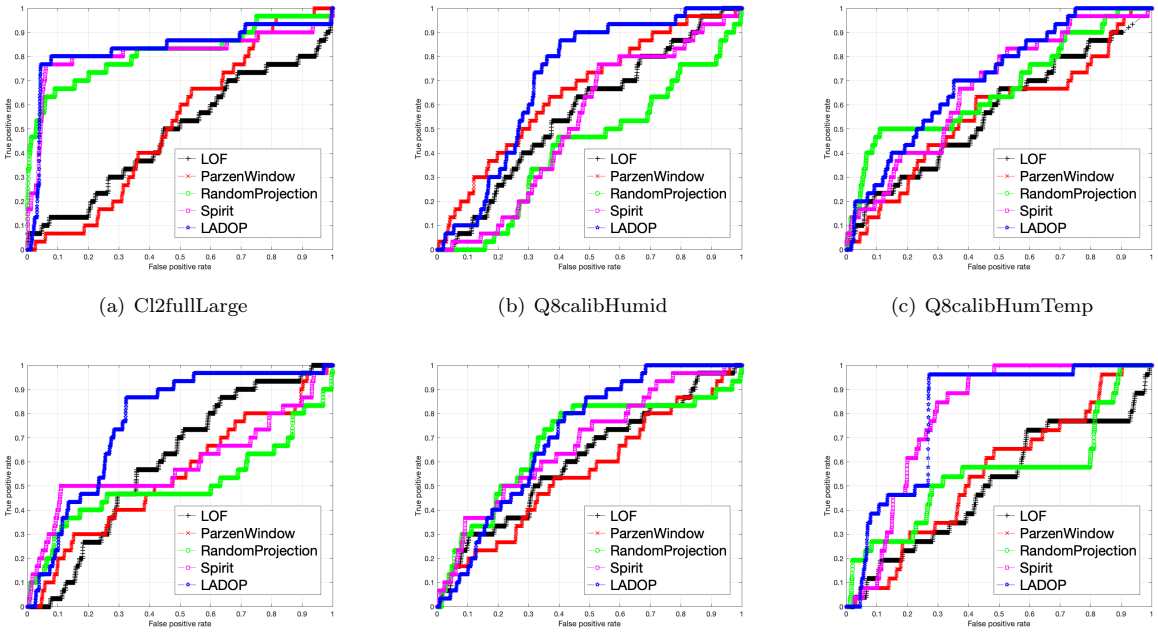
[Agg13] Charu C. Aggarwal. *Outlier Analysis*. Springer Publishing Company, Incorporated, 2013.

[AS12] Charu C. Aggarwal and Karthik Subbian. Event detection in social streams. In *SIAM*, pages 624–635, 2012.

[AY01] Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. In *ACM SIGMOD*, pages 37–46, 2001.

	Dimensions		Algorithmes				
	n	p	**LOF**	**Parzen Window**	**Random Proj**	**SPIRIT**	**LADOP**
Cl2fullLarge	4310	166	48.4085	52.2189	81.6773	81.5499	**83.2772**
Q8calibHumid	7712	48	56.9759	65.6476	42.8382	52.6173	**71.0079**
Q8calibHumTemp	7712	56	56.9299	56.0494	66.6397	66.8294	**70.7542**
Q8calibLight	7712	48	60.2155	54.4726	50.6242	59.3872	**76.6588**
Q8calibVolt	7712	46	61.0887	56.0314	67.9746	67.6319	**71.0696**
Occupancy	2782	5	50.5123	55.3101	55.1365	79.3278	**80.6133**

TABLE 1 – AUC des méthodes sur les jeux de données utilisés.



(a) Cl2fullLarge (b) Q8calibHumid (c) Q8calibHumTemp

(d) Q8calibLight (e) Q8calibVolt (f) Occupancy

FIGURE 3 – Courbes ROC des différentes approches.

[BBC18] Seif-Eddine Benkabou, Khalid Benabdeslem, and Bruno Canitia. Unsupervised outlier detection for time series by entropy and dynamic time warping. *KAIS*, 54(2) :463–486, 2 2018.

[BKNS00] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof : Identifying density-based local outliers. SIGMOD '00, pages 93–104, New York, NY, USA, 2000. ACM.

[BSAT06] Suratna Budalakoti, Ashok Srivastava, R Akella, and Eugene Turkov. Anomaly detection in large sets of high-dimensional symbol sequences. Technical Report NASA TM-2006-214553, 01 2006.

[BSO+09] Suratna Budalakoti, Ashok N Srivastava, Matthew Eric Otey, et al. Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS PART C, APPLICATIons and reviews"*, 39(1) :101, 2009.

[CBK09] Varun Chandola, Arindam Banerjee, and

Vipin Kumar. Anomaly detection : A survey. *ACM Comput. Surv.*, 41(3) :15 :1–15 :58, 2009.

[CCJY08] Haifeng Chen, Haibin Cheng, Guofei Jiang, and Kenji Yoshihira. Exploiting local and global invariants for the management of large scale information systems. In *(ICDM 2008)*, pages 113–122, 2008.

[CMK08] Varun Chandola, Varun Mithal, and Vipin Kumar. A comparative evaluation of anomaly detection techniques for sequence data. In *(ICDM*, pages 743–748, 2008.

[EBES05] Paul F. Evangelista, Piero Bonnisone, Mark J. Embrechts, and Boleslaw K. Szymanski. Fuzzy roc curves for the 1 class svm : Application to intrusion detection. *IJCNN*, 07 2005.

[GPST06] Pedro Galeano, Daniel Peña, and Ruey S. Tsay. Outlier detection in multivariate time series by projection pursuit. *JASA*, 101 :654–669, 06 2006.

[Haw80] D.M Hawkins. *Identification of Outliers*. Chapman and Hall, London Ð New York, 1980.

[Hig02] Nicholas J Higham. *Accuracy and stability of numerical algorithms*, volume 80. Siam, 2002.

[HT18] Madelon Hulsebos and David Tax. Outlier detection in multivariate time series exploiting reconstructions from random projections. 2018.

[KLF05] E. Keogh, J. Lin, and A. Fu. Hot sax : efficiently finding the most unusual time series subsequence. In *(ICDM'05)*, pages 8 pp.–, Nov 2005.

[KS13] Emrah Kılıç and Pante Stănică. The inverse of banded matrices. *Journal of Computational and Applied Mathematics*, 237 :126–135, 01 2013.

[LDHL17] Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. Radar : Residual analysis for anomaly detection in attributed networks. In *IJCAI*, pages 2152–2158. ijcai.org, 2017.

[NHCD10] Feiping Nie, Heng Huang, Xiao Cai, and Chris H. Q. Ding. Efficient and robust feature selection via joint $\ell_{2,1}$ -norms minimization. In *NIPS*, pages 1813–1821. Curran Associates, Inc., 2010.

[PES01] Leonid Portnoy, Eleazar Eskin, and Salvatore Stolfo. Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, 11 2001.

[PSF05] Spiros Papadimitriou, Jimeng Sun, and Christos Faloutsos. Streaming pattern discovery in multiple time-series. VLDB '05, pages 697–708. VLDB Endowment, 2005.

[SC05] Stan Salvador and Philip Chan. Learning states and rules for detecting anomalies in time series. *Applied Intelligence*, 23(3) :241–255, Dec 2005.

[SO11] Yiyuan She and Art B. Owen. Outlier detection using nonconvex penalized regression. *Journal of the American Statistical Association*, 106(494) :626–639, 2011.

[TL13] Jiliang Tang and Huan Liu. Coselect : Feature selection with instance selection for social media data. In *SDM*, pages 695–703. SIAM, 2013.

[YBT06] Kai Yu, Jinbo Bi, and Volker Tresp. Active learning via transductive experimental design. ICML '06, pages 1081–1088, New York, NY, USA, 2006. ACM.

[YC02] Dit-Yan Yeung and C. Chow. Parzen-window network intrusion detectors. In *Object recognition supported by user interaction for service robots*, volume 4, pages 385–388 vol.4, 2002.

Approche GAN pour la génération d'images sous contraintes de pixels

Cyprien Ruffino[1], Romain Hérault[1], Eric Laloy[2], Gilles Gasso[1] *

1- Normandie Univ, UNIROUEN, UNIHAVRE, INSA Rouen, LITIS
76 000 Rouen, France
2- Belgian Nuclear Research, Institute Environment, Health and Safety,
Boeretang 200 - BE-2400 Mol, Belgium

April 10, 2019

Abstract

Les réseaux génératifs aversaires (GAN) ont montré leur efficacité pour des tâches de génération d'images de manière non supervisée ces dernières années. Plusieurs travaux ont étendu les GANs à la complétion (*inpainting*) d'images en conditionnant la génération à des parties de l'image que l'on veut reconstruire. Ces méthodes présentennt des limitations dans les configurations où seul un très petit sous-ensemble des pixels de l'image est connu à l'avance. Dans cet article, nous étudions l'efficacité du conditionnement des GANs en ajoutant un terme de régularisation explicite pour prendre en compte des contraintes sur la valeur de quelques pixels de l'image (entre 0.5 et 1% des pixels sont imposés). L'influence de ce terme de régularisation sur la qualité des images générées et la satisfaction des contraintes est également analysée. Des expériences menées sur MNIST et FashionMNIST montrent que ce terme de régularisation permet de contrôler le compromis entre la qualité des images générées et la satisfaction des contraintes.

1 Introduction

In this work we consider an extreme setting of inpainting task: we assume that only a few pixels, less than a percent of the considered image size, are known and that these pixels are randomly scattered across the image (see Fig.1c). This raises the challenge of how to take advantage of this scarce and unstructured a priori information to generate high quality images. Besides methodological novelty, a method that can tackle this problem would find applications to GAN-based geostatistical simulation and inversion in the geosciences [LHJL18].

More specifically, this paper proposes an extension of the Conditional Generative Adversarial Network (CGAN) [MO14] framework to learn the distribution of the training images given the constraints (the known pixels). To make the generated images honoring the prescribed pixel values, we use a regularization term measuring the distance between the real constraints and their generated counterparts. Thereon we derive a learning scheme and analyze the influence of the used regularization term on both the quality of the generated images and the fulfillment of the constraints. By experimenting with a wide range of values for the additional hyper-parameter introduced by the regularization term, we show for the MNIST [LBBH98] and FashionMNIST [XRV17] datasets that our approach is effective and allows for controlling the trade-off between the quality of the generated samples and the satisfaction of the constraints.

2 Related works

Generative Adversarial Networks [GPAM+14] basically consist of an algorithm for training generative models in an unsupervised way. It relies on a game between a generator, G, and a discriminator network, D, in which G learns to produce new data with similar spatial characteristics/patterns as in the true data while D learns to distinguish real examples from generated ones. Training GANs is equivalent to finding a Nash equilibrium to the following mini-max game:

*This research was supported by the CNRS PEPS I3A REG-GAN project and the ANR-16-CE23-0006 grant *Deep in France*

(a) Original Image (b) Regular Inpainting (c) Scatter Inpainting (d) Generated Image

Figure 1: Difference between regular inpainting (b) and the problem undertaken in this work (c). The image obtained with our framework is shown in (d).

$$\min_G \max_D L(D,G) = \mathop{\mathbb{E}}_{x\sim P_r}\Big[\log(D(x))\Big] + \mathop{\mathbb{E}}_{z\sim P_z}\Big[\log(1-D(G(z)))\Big] \quad (1)$$

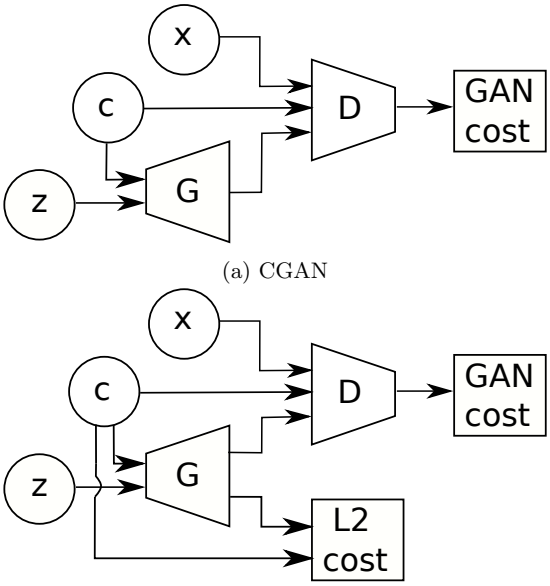
where P_z is a known distribution, usually normal or uniform, in which latent variables are drawn, and P_r is the distribution of the real samples.

Yeh et al. [YCL+17] introduced an inpainting method which consists of taking a pre-trained generator and exploring its latent space $\mathcal{Z}$ via gradient descent, to find a latent vector, z, which induces an image close to the altered one while its quality remains close to the real samples. This method was applied by Mosser et al. [MDB18] for 3D image completion with few constraints. However, the location of the constraints in their approach was fixed, instead of randomly scattered.

Some other approaches rely on Conditional Generative Adversarial Networks (CGAN) [MO14]. This is a variant of GANs in which additional information, c, is given to both the generator and the discriminator as an input (see Fig.2a). The optimization problem becomes:

$$\min_G \max_D L(D,G) = \mathop{\mathbb{E}}_{\substack{x\sim P_r\\ \tilde{c}\sim P_{c|x}}}\Big[\log(D(x,\tilde{c}))\Big] + \mathop{\mathbb{E}}_{\substack{z\sim P_z\\ c\sim P_c}}\Big[\log(1-D(G(z,c),c))\Big] \quad (2)$$

In it seminal version [MO14], CGANs are used for class-conditioned image generation by giving the labels of the images to the networks. However, several kind of conditioning data can be used even a full image to do image-to-image translation [IZZE17] or image inpainting [YLY+18, DU18].



(a) CGAN

(b) Our approach

Figure 2: Different GAN Setups

3 Proposed approach

In this work, we retain the CGAN approach and add a reconstruction loss term to further enforce the prescribed pixel values (usually less than a percent of the image). With this setup, the generator can be used to generate images from constraints unseen during the training.

Given a learning set of images $X \in [-1,1]^{P\times P}$ drawn from an unknown distribution P_r and a sparse matrix $C \in [-1,1]^{P\times P}$ as the given constrained pixels, the problem we focused on consists in finding a generative model G with input $z \sim P_z$, a random vector sampled from a known distribution, and constrained pixel values $\tilde{C} \in [-1,1]^{P\times P}$ that could generate an image satisfying the constraints while likely following the distribution P_r. Enforcing the constraints in the CGAN

(a) Original Image (b) Constraints (c) Generated Image (d) Satisfied Consts.

Figure 3: Generation of a sample during training. We first sample an image from a training set (a) and we sample the constraints from it. Then our GAN generates a sample (c). The constraints with squared error smaller than $\epsilon = 0.1$ are deemed satisfied and shown by green pixels in (d) while the red pixels are unsatisfied.

framework leads to the following problem:

$$\min_G \max_D L(D,G) = \mathop{\mathbb{E}}_{\substack{X\sim P_r \\ \tilde{C}\sim P_{C|X}}} \Big[\log(D(X,\tilde{C}))\Big] + \mathop{\mathbb{E}}_{\substack{z\sim P_z \\ C\sim P_C}} \Big[\log(1-D(G(z,C),C))\Big]$$
$$\text{s.c. } C = M(C) \odot G(z,C)) \quad (3)$$

where $\odot$ is the Hadamard (or point-wise) product and $M(C)$ is a corresponding masking matrix. $M(C)$ is a sparse matrix with entries equal to one at constrained pixels location. As the equality constraint in 3 is hard to enforce during training, we rather investigate a relaxed version of the problem. Indeed, we minimize the L_2 norm between the constrained pixels and the generated values (see Fig.2b). The objective function, with $\lambda \geq 0$ a regularization parameter, becomes:

$$\begin{aligned} L(D,G) = & \mathop{\mathbb{E}}_{\substack{X\sim P_r \\ \tilde{C}\sim P_{C|X}}} \Big[\log(D(X,\tilde{C}))\Big] \\ & + \mathop{\mathbb{E}}_{\substack{z\sim P_z \\ C\sim P_C}} \Big[\log(1-D(G(z,C),C))\Big] \\ & + \mathop{\mathbb{E}}_{\substack{z'\sim P_z \\ 'C\sim P_C}} \Big[\lambda \left\|C' - M(C') \odot G(z,C')\right\|_2^2\Big] \end{aligned} \quad (4)$$

4 Experiments

We experiment on the MNIST [LBBH98] and Fashion-MNIST [XRV17] datasets, which consist of images of size 28×28px. We split the official training set into a new training set (90%) and a validation set (10%). The official test set remains our test set. A fifth of each so defined set is used to generated the matrix of constraints C by randomly selecting 0.5% of the pixels. These images are then removed from the training sets, to avoid correlation between real example presented to the discriminator and constrained maps given to the generator.

A discriminator such as presented in DCGAN [RMC15] has been chosen with only two convolutional layers of 64 and 128 filters, Leaky ReLU activations and batch normalization [IS15]. For the generator we retain the DCGAN architecture with a fully-connected layer and two transposed convolutional layers of 128 and 64 filters with ReLU activations and batch normalization. An example of a generated image with the corresponding constraints can be seen in figures 3c and 3d.

We evaluate our models based on both the satisfaction of the constraints and the visual quality of the generated samples. On one hand, we use the mean squared error between the provided constrained values and the constrained pixels in the generated image. On the other hand, evaluating the visual quality of an image is not a trivial task [TOB15]. However, the recently developed metric referred to as Fréchet Inception Distance (FID) [HRU+17] seems to be a good metric of performance. Since using the FID requires a pre-trained classifier, we trained a simple convnet with MNIST/FashionMNIST labels as target. Lower layers of the classifier are then used to produce high-level features needed by the distance:

$$FID = ||\mu_r - \mu_g||2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}), \quad (5)$$

where μ_r, Σ_r, μ_g and Σ_g are the mean and the covariance matrices of extracted features obtained on respectively the real and the generated data. To overcome classical GANs instability, the networks are trained 10 times and the median of the best scores on the test set at the best epoch are recorded. The epoch that minimizes $\sqrt{FID^2 + MSE}$ on the validation set is considered as the best epoch.

Empirical evidences show that with a good choice of λ, the regularization term helps the generator to learn enforcing the constraints (Fig.4), leading to smaller MSEs than when using the CGAN approach only ($\lambda = 0$) and with minor detrimental effects on the quality

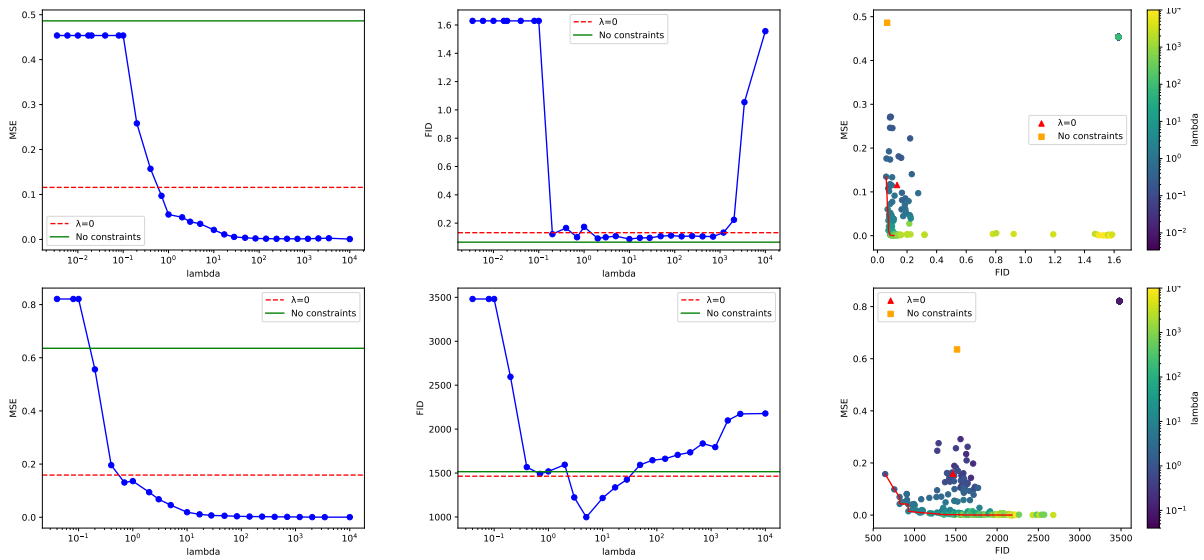


Figure 4: MSE (left) and FID (center) w.r.t. the regularization parameter λ; MSE w.r.t the FID (right). Dataset MNIST (top), Fashion MNIST (bottom).

of the samples (Fig.4). For Fashion MNIST, the regularization term even leads to a better image quality compared to the quality provided by GAN and CGAN approaches. Fig. 4 illustrates that the trade-off between image quality and the satisfaction of the constraints can be controlled by appropriately setting the value of λ. Nevertheless, for small values of λ the GAN fails to learn and only generates completely black samples. This leads to the plateaus seen for both the MSE and the FID in Fig. 4.

5 Conclusion

In this paper, we investigate the effectiveness of adding a regularization term to the conditioning of GANs to deal with cases where only a small subset of the image one wants to generate is known beforehand. Empirical evidences illustrate that the proposed framework helps obtaining good image quality while best fulfilling the constraints compared to classical GAN approaches. In future work, we plan to extend this study to GAN conditioning in situations where no trivial mapping exists between the conditions and the generated samples, such as class-wise conditioning or more structured conditions.

References

[DU18] Ugur Demir and Gozde Unal. Patch-based image inpainting with generative adversarial networks. *arXiv preprint arXiv:1803.07422*, 2018.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[HRU+17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[IZZE17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2017.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LHJL18] Eric Laloy, Romain Hérault, Diederik Jacques, and Niklas Linde. Training-image based geostatistical

inversion using a spatial generative adversarial neural network. *Water Resources Research*, 54(1):381–406, 2018.

[MDB18] Lukas Mosser, Olivier Dubrule, and Martin J Blunt. Conditioning of three-dimensional generative adversarial networks for pore and reservoir-scale models. *arXiv preprint arXiv:1802.05622*, 2018.

[MO14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[TOB15] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[YCL+17] Raymond A Yeh, Chen Chen, Teck-Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *CVPR*, volume 2, page 4, 2017.

[YLY+18] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. *arXiv preprint arXiv:1801.07892*, 2018.