Learning similarities for linear classification : theoretical foundations and algorithms

Maria-Irina Nicolae

École Doctorale ED488 "Sciences, Ingénierie, Santé"

Learning Similarities for Linear Classification: Theoretical Foundations and Algorithms

---

Apprentissage de similarités pour la classification linéaire : fondements théoriques et algorithmes

Thèse préparée par **Maria-Irina Nicolae** pour obtenir le grade de :
Docteur de l'Université Jean Monnet de Saint-Etienne
Domaine: **Informatique**

Soutenance le 2 Décembre 2016 au Laboratoire Hubert Curien
devant le jury composé de :

Maria-Florina Balcan	Maître de conférences, Carnegie Mellon University	Examinatrice
Antoine Cornuéjols	Professeur, AgroParisTech	Rapporteur
Ludovic Denoyer	Professeur, Université Pierre et Marie Curie	Rapporteur
Éric Gaussier	Professeur, Université de Grenoble-Alpes	Co-directeur
Amaury Habrard	Professeur, Université de Saint-Etienne	Examinateur
Liva Ralaivola	Professeur, Aix-Marseille Université	Examinateur
Marc Sebban	Professeur, Université de Saint-Etienne	Directeur

---

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of machine learning is to design accurate prediction algorithms based on available training data. This data can be seen as experience from which one can learn by example: the previous instances have to be taken into account when making decisions. The process is different from memory-based systems which only recognize past instances. In learning, the purpose is to determine the model associated with the data and be able to apply it on unseen instances. In most cases, once the modeling step is performed, the initial data can be discarded, and the model can predict outcomes for new data arriving in the system. In contrast to memory-based systems, this implies that machine learning has generalization capacities.

Learning algorithms are successfully used in a variety of applications, including:

- Computer vision (e.g. image segmentation, face detection, object recognition);
- Signal processing (e.g. speech recognition and synthesis, voice identification);
- Information retrieval (e.g. search engines, recommender systems);
- Unassisted vehicles (e.g. robots, drones, self-driven cars);
- Computational linguistics;
- Computational biology and genetics;
- Medical diagnosis.

This list is not a comprehensive one, and new applications for learning algorithms are designed every day.

Data can be made available to the algorithm in different forms. Feature vectors are often used to contain categorical or numerical information, like the location of a house, its price, its surface and so on. Structured data is used to represent information that cannot be directly stored in the previous form, such as strings (e.g. text documents), trees (e.g. XML content) or graphs (e.g. social networks). Another type of structured data present in numerous applications are time series, which follow the evolution of a process across time (e.g. stock values, patient vital signs or weather information). When presented with any of the previous types of data (or a mix of them), the algorithm can be asked to perform a certain number of learning tasks. In *supervised learning*, the data comes with

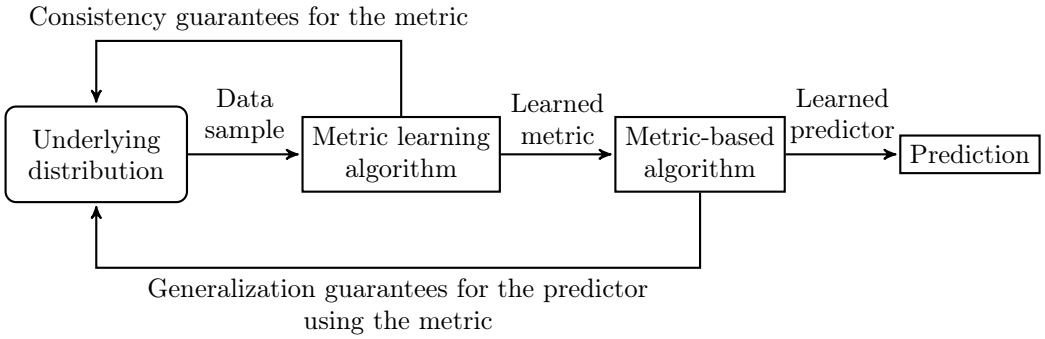


Figure 1.1: The supervised machine learning workflow. The question of the generalization capacity can be asked at two levels: the metric and the predictor using it.

a label associated to each instance. The algorithm must learn to predict these labels for unseen examples. When the label comes from a discrete set of values (e.g. the gender of a person), the task is called *classification*. Conversely, when labels are continuous (e.g. prices), the task to be performed is *regression*. We will mainly focus on supervised tasks in this document. *Unsupervised learning* covers the case where no label information is available for the data. In this context, the algorithm is asked to perform a task for which label information is not necessary. For example, *clustering* tries to find an inherent pattern in the data which allows to divide examples into groups named clusters. *Semi-supervised learning* covers similar tasks to the supervised setting, but is able to additionally integrate unlabeled samples.

The performance of a learning algorithm is strongly related to the quantity and quality of data available. As in the case of all algorithms, performance measures for learning methods include time and space complexity. While computational efficiency is a significant concern in machine learning, more specific measures allow evaluating algorithms from a learning perspective. The most important of these is the quality of the prediction. An additional particular measure is *sample complexity*, which indicates the quantity of data required by an algorithm to learn to solve a certain task. More precisely, theoretical guarantees can be provided for the performance of a learning algorithm with respect to the quantity of data available and a notion of complexity of the chosen model. These types of guarantees can be derived using methods from statistical learning theory.

Most machine learning algorithms are brought to compare data instances, either for constructing the model or making predictions. Comparisons are performed using a notion of metric that gives a measure of resemblance between examples. This is true for supervised methods, like k-nearest neighbors or support vector machines, as well as for unsupervised ones, such as K-means. However, choosing an adapted metric can prove to be a difficult task, and the performance of the algorithm strongly depends on it. One hopes that the measure discriminates well the data for the given task (e.g. for classification, recognizing instances of the same label as similar). Moreover, standard metrics, such as the Euclidean distance, do not have the capacity to adapt to a task or to incorporate semantic constraints, thus sometimes missing the relevant information.

Metric learning aims at solving exactly this problem by automatically learning custom metrics from data. This step is performed prior to applying another machine learning algorithm using the learned metric to solve a specific task. Learning metrics for the purpose of classification is the main subject that we address in this thesis. Metric learning can be seen as part of a larger topic termed *representation learning*: under the new metric, the representation used for the data changes. Representation learning also includes fields such as feature learning, deep learning, kernel learning and many others. Metric learning has received a lot of attention over the past fifteen years, especially for the supervised setting. Most methods learn the parameters of a Mahalanobis distance under constraints induced by the data, which is equivalent to changing the representation of the instances under a linear transformation. In the supervised case, the learned metric has the objective to represent the instances with the same or close (respectively different) labels as similar (respectively dissimilar). State of the art methods for feature vectors are more scalable and efficient in computation than previous approaches. On the other hand, metric learning for time series applications has received scarce attention, most likely because of the more complex structure of the data. Time series coming from real applications usually have different lengths, sampling rates and phases. For this reason, in order to compare two time series, one must first align them. Roughly speaking, this implies finding all the correspondences between time moments. Dynamic time warping (DTW) is the most well-known method for providing the optimal alignment, which is based on a cost matrix. Metric learning for time series usually aims at optimizing this cost under constraints deducted from the examples.

The current advancement of the metric learning field suffers from a few non-negligible limitations. The most important is probably the lack of theoretical analyses for most existing methods. To be more precise, one would want to evaluate the generalization capacity associated to learning a metric. For the particular case of metric learning, this notion can be declined under two aspects (see Figure 1.1). First, the metric should be consistent with the data, that is it should be able to generalize well outside the training data, on unseen examples from the same distribution. This type of guarantee has only been provided for a small number of methods. Second, the learned metric should be related to the algorithm that is using it, improving its performance with respect to standard metrics. For example, learning a metric from local constraints for a local classification rule might not perform well when used with a global linear classifier. This second type of guarantee is also rare in the field, and, even though some methods improve a criterion related to the classifier, this link is not formally established. In practice, the learned metric is plugged in the classifier in hope of making if better. The theory of (ϵ, γ, τ)-good similarity functions has been one of the first results trying to relate the properties of a similarity function to the performance that can be expected of it in classification. We describe this framework in detail later in the thesis, as our contributions make use of it. A second limitation in metric learning comes from the fact that most methods work with metrics that enforce distance properties. Satisfying this type of constraints can be expensive even when using modern numerical optimization techniques, which increases the overhead of performing a metric learning step. This limitation joint with the lack of guarantees for performance amelioration implies that the effort spent in metric learning might actually result in an inadequate metric.

In this thesis, we aim to address the previously mentioned limitations. All our contributions are based on learning similarity functions, a different family of metrics which are less constraint than distances and constitute a promising, but less explored avenue for research. We also derive generalization guarantees for the learned metric and the associated classifier for all the methods we propose. First, we introduce a general framework for performing similarity and linear classifier learning simultaneously. This setting is designed for feature vectors and works for a wide range of similarity functions. Moreover, the approach is semi-supervised, which allows it to leverage unlabeled examples in order to improve performance. Our following contribution allows us to tackle the problem of metric learning for multivariate time series. We propose to learn a bilinear similarity function to be used for linear classification. The similarity uses the optimal alignments and allows to better reweigh the features.

Context of this work The work for this thesis was performed in machine learning teams from two establishments: the Data Intelligence group of Laboratoire Hubert Curien UMR CNRS 5516, part of University of Saint-Étienne and University of Lyon, and the Data Analysis, Modeling and Machine Learning (AMA) group of Laboratoire Informatique de Grenoble, part of Grenoble Alps University. Funding for this project was provided by a grant from Région Rhône-Alpes.

Outline of the thesis This dissertation is organized as follows.

- Chapter 2 introduces the notions related to statistical machine learning that are necessary for the rest of this document. We begin with a formal presentation of supervised learning, then give some examples of classic learning algorithms, followed by the learning theory frameworks for deriving generalization guarantees.
- Chapter 3 is dedicated to metrics in general, and metric learning in particular. After presenting a number of standard metrics for feature vectors and time series, we provide a comprehensive survey of (semi-)supervised metric learning for these types of data. This chapter explains in more detail the limitations that have determined the direction of our contributions.
- Chapter 4 presents our contributions on metric learning for feature vectors. We introduce a general framework called JSL (Joint Similarity Learning) for learning a similarity function and a global linear classifier simultaneously. JSL is capable of integrating unlabeled information in the form of landmarks based on which global constraints are generated. This property allows our framework to adapt to the particular setting in which only a small amount of labeled data is available. The formulation optimizes the (ϵ, γ, τ)-goodness of the similarity, which ensures its performance in classification. JSL is convex and can be solved efficiently under mild constraints over the choice of the similarity function. We derive theoretical guarantees for JSL using two different frameworks: the algorithmic robustness and the uniform convergence using Rademacher complexity. The experiments compare JSL to a large range of state of the art methods and prove its efficiency.

- Chapter 5 gathers our contributions on metric learning for temporal data by introducing SLTS (Similarity Learning for Time Series). We propose to learn a bilinear similarity function for multivariate time series based on the optimal alignment. The learned similarity is to be used in a classification task to determine a linear separator. The problem we solve is convex and easy to solve. Making use of the uniform stability framework, we provide the first theoretical guarantees in the form of a consistency bound for the learned similarity. Moreover, improving the (ϵ, γ, τ)-goodness of the metric provides additional guarantees concerning its performance for linear classification. The experimental study shows that the proposed approach is efficient, while yielding sparse classifiers.

- Chapter 6 concludes our work and discusses avenues for possible future work.

Notation Table 1.1 explains the notations used throughout this thesis.

Notation	Description
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^d$	Set of d-dimensional real-valued vectors
$\mathbb{R}^{d \times d'}$	Set of $d \times d'$ real-valued matrices
$\mathbb{S}_+^d$	Cone of symmetric PSD $d \times d$ real-valued matrices
$\mathcal{S}$	An arbitrary set
$\lvert\mathcal{S}\rvert$	The cardinality of $\mathcal{S}$
$\mathcal{S}^n$	A set of n elements from $\mathcal{S}$
$\mathcal{X}$	Input space
$\mathcal{Y}$	Output space
$z = (x, y) \in \mathcal{X} \times \mathcal{Y}$	A labeled instance
$\mathbf{x}$	An arbitrary vector
x_i	The i^{th} component of $\mathbf{x}$
$\mathbf{M}$	An arbitrary matrix
$\mathbf{I}$	The identity matrix
$M_{i,j}$	Entry at row i and column j of matrix $\mathbf{M}$
$[\cdot]_+$	The hinge function
$\lvert\lvert \cdot \rvert\rvert$	Arbitrary norm
$\lvert\lvert \cdot \rvert\rvert_p$	L_p norm
A	An arbitrary time series
$x \sim P$	x is drawn i.i.d. from probability distribution P
$\Pr[\cdot]$	Probability of an event
$\mathbb{E}[\cdot]$	The expectation of a random variable

Table 1.1: Summary of notation.

Chapter 2

Fundamentals of Theoretical Learning

Chapter abstract

This chapter introduces the most common learning setups, focusing on the standard setting for supervised learning, as well as some fundamental notions from statistical learning theory. We present the main analytical frameworks and tools for deriving generalization guarantees. We begin with the Probably Approximately Correct (PAC) learning model, then present multiple frameworks for deriving PAC generalization bounds: uniform convergence with different measures of complexity, uniform stability and algorithmic robustness.

2.1 Introduction

The practical objective of machine learning is to make correct predictions for items that were not seen before in an efficient and robust way. This is done based on a model inferred from available data instances.

A classic setting for this type of problem is *supervised learning* (Bishop, 2006). Its particularity is that the available data for determining the model is labeled, i.e. the items are annotated with target values similar to those which the algorithm should predict. When the annotation comes from a set of discrete categories, and the purpose is to assign a category to each new item, the problem is called *classification*. In contrast, if the output is a continuous, usually real-valued variable, the learning task is called *regression*.

A different setting is *unsupervised learning*, where the training data is a set of items with no target annotation. Here, the goal can be to discover a pattern that allows to aggregate the data in similar groups in the case of *clustering* (Kaufman & Rousseeuw, 1990), to determine the distribution of the data for *density estimation* (Silverman, 1986), or to perform data *visualization* in two or three dimensions by projecting it to such a low dimensional space (Post et al., 2002).

Semi-supervised learning (Chapelle et al., 2006) covers the same tasks as the supervised setting when only part of the training data is annotated. The purpose is to extract

additional information (typically, about their underlying statistical distribution) from the unlabeled data in order to perform better than based only on the labeled sample. This is usually done in a transductive setting, by label propagating onto close unlabeled examples (Vapnik, 1998), or by integrating the structure of unlabeled neighborhoods through graph nodes adjacency information (Zhu, 2005).

Other learning paradigms that are not in the scope of this document include transfer learning (Pan & Yang, 2010), which aims at better solving one learning problem by using information from another related problem, and reinforcement learning (Sutton & Barto, 1998), where the algorithm takes sequential actions to maximize a notion of reward, without being presented with an explicit access to correct results.

This dissertation focuses on (semi-)supervised learning, more precisely classification, which we introduce formally in the following section.

2.2 Supervised Learning

In a supervised setting, the learning algorithm has access to a labeled training set coming from a fixed but unknown distribution, used to create a model with prediction capacities. Let us define these notions formally.

Domain set An arbitrary set $\mathcal{X}$. Usually, domain points are vectors of features coming from $\mathcal{X} \subseteq \mathbb{R}^d$ and are referred to as instances. The *features* or the characteristics of the data have to be informative to guide the learning algorithm effectively. The choice of features is left to the user and reflects his prior knowledge about the task, but a lot of research effort has been put into making this selection automatically (Guyon, 2003).

Label set Also known as annotations or target values, these are the feedback provided to the learning algorithm. In the classification setting, $\mathcal{Y}$ represents the set of all possible labels. For the purpose of this discussion, and without loss of generality, we will focus on binary labels, usually $\{-1, 1\}$.

Training data A training sample $\mathcal{S}$ is a set $\{z_i = (\mathbf{x}_i, y_i)\}_{i=1}^n$ of size n independent instances, identically distributed (i.i.d.) according to an unknown distribution P over the space of instances and labels $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Such labeled examples are often called training set and are the data based on which the algorithm learns.

Output We assume there exists a correct *labeling function* $f : \mathcal{X} \to \mathcal{Y}$ for which $f(x) = y$ for all (x, y) drawn from P. The labeling function is unknown to the algorithm. Following the learning process, the algorithm provides a function $h : \mathcal{X} \to \mathcal{Y}'$ coming from a hypothesis space H that best predicts the behavior of f. For instance, H can be the family of all linear separators (lines) in a plane. Note that $\mathcal{Y}'$ can be different from the label space $\mathcal{Y}$. h is also called a predictor, a hypothesis (or a classifier in classification), and is used to predict the labels of new instances arriving. We will denote by $h_{\mathcal{S}}$ the model obtained from sample $\mathcal{S}$. In many types of algorithms, the

training data is not necessary anymore once the model is built, and prediction is done only based on the model h.

Throughout this dissertation, we will consider two types of data: feature vectors coming from $\mathcal{X} \subseteq \mathbb{R}^d$ and time series defined as follows.

Definition 2.1 (Time series)**.** *A time series is a sequence of values of a quantity obtained at successive times. The term univariate time series refers to a time series that consists of single (scalar) observations recorded sequentially over time increments, i.e.* $\mathcal{X} \subseteq \mathbb{R}$*. A multivariate time series is a time series that records multiple values at each time increment, i.e. where each time moment is a vector of features, and* $\mathcal{X} \subseteq \mathbb{R}^d$*.*

More formally, we can define supervised learning in the following way.

Definition 2.2 (Supervised learning)**.** *Supervised learning is the task of finding a modeling function* $h : \mathcal{X} \to \mathcal{Y}'$ *coming from a hypothesis class* H *that best predicts the value of* y *given* x *for any* (x, y) *drawn from* P*.*

As stated before, for a model to be of good quality, its predictions should match the true labels of the data. When choosing the best hypothesis to fit the data, its adequacy is measured through the use of a loss function $\ell : H \times \mathcal{Z} \to \mathbb{R}^+$. Loss functions are nonnegative and usually take value zero (or close to zero) when the prediction is correct and higher values otherwise. We define the *true risk* as the measure over P of the quality of a hypothesis h under ℓ.

Definition 2.3 (True risk)**.** *Given a hypothesis* $h \in H$*, the true risk* R^ℓ_P *of* h *with respect to a loss function* ℓ *is the expected loss suffered by* h *on the distribution* P*:*

$$R^\ell_P(h) = \mathbb{E}_{z\sim P}[\ell(h, z)].$$

The true risk is also known under the names of generalization error, risk or true error, which we use interchangeably throughout this document. Intuitively, it measures the capacity of h to make correct predictions for all instances $(x, y) \in P$. The goal of supervised learning is to find the hypothesis h that obtains the lowest true risk. However, since the risk is an expected true value and depends on the unknown distribution P, it cannot be computed directly. The learner should thus minimize a substitute value instead. This is usually the empirical value of the risk on the available sample $\mathcal{S}$, also known as the *empirical risk*.

Definition 2.4 (Empirical risk)**.** *Given a hypothesis* $h \in H$ *and a sample* $\mathcal{S} = \{z_i\}^n_{i=1}$ *of size* n*, the empirical risk (or empirical error)* $R^\ell_\mathcal{S}(h)$ *with respect to a loss function* ℓ *is the average loss incurred by the algorithm on the instances of* $\mathcal{S}$*:*

$$R^\ell_\mathcal{S}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h, z_i).$$

When using the previous notations, we omit the loss function ℓ when it is clear from the context.

The predictor h can take different forms, depending on the family of hypotheses learned: it can be a vector (usually from $\mathbb{R}^p$), a matrix (from $\mathbb{R}^{p \times p'}$), the parameters of a neural network, etc. For binary classification, h generally outputs a scalar in $\mathbb{R}$. In this case, the most intuitive loss function is the zero-one loss, which tells for each instance if it has been labeled correctly:

$$\ell_{0/1}(h, z) = \begin{cases} 1 & \text{if } yh(\mathbf{x}) < 0, \\ 0 & \text{otherwise.} \end{cases}$$

The product $yh(\mathbf{x})$, which represents the margin, determines if y and $h(\mathbf{x})$ agree in sign; the error is thus 1 when this is not the case, computing the overall proportion of incorrect predictions. Even though this loss function is arguably the most adequate for binary classification, its mathematical properties make it hard to optimize, as we will see later.

In the next section we describe the most well known learning frameworks that use the current setting to provide a good model.

2.3 Learning Good Hypotheses

In this section, we give an intuition about what makes a good hypothesis and the most well known methods to learn one.

According to the law of large numbers, the empirical risk asymptotically converges to the true risk when the size of the sample is infinite. In practice, the quantity of data is always limited and sometimes rather small. Moreover, choosing a hypothesis class complex enough (e.g. a high degree polynomial) will allow to perfectly fit the training sample and artificially minimize the empirical error without also reducing the true risk. The result is a predictor with low generalization capacity. The phenomenon of conserving an elevated true risk in spite of a low sample error is called *overfitting* and is the result of a too complex hypothesis h or insufficient data available lying in high dimensional space (curse of dimensionality). In this case, the model is too sensitive to small fluctuations in the training set, adapting to random noise. This behavior is also called high variance, in reference to the variance of the obtained estimator.

In contrast to overfitting, *underfitting* occurs when the algorithm is unable to learn the relevant relations between data and output. This behavior is usually induced by erroneous assumptions in the algorithm, i.e. the hypothesis class is not complex enough to justify the data. For example, a linear classifier will not be able to separate instances that are not linearly separable w.r.t. to P. This phenomenon is also called high bias, referring to the fact that the hypothesis is biased towards the assumptions that were made about the problem, instead of adapting to the sample.

Both elevated variance and bias can prevent a learning algorithm from generalizing well. The *bias–variance trade-off* is the problem of finding the best compromise between these two sources of error: choosing a hypothesis that is just as complex as needed to model the data. Usually, the higher the variance, the smaller the bias, and vice-versa.

We now proceed to presenting the classic strategies for obtaining good hypotheses: mini-

mizing the risk via the empirical error.

Empirical Risk Minimization Since the training sample is the only information available to the algorithm, minimizing the empirical risk on this data seems like a natural way to obtain a good hypothesis. This strategy is called *empirical risk minimization* (ERM) and is formulated as follows:

$$h_{\mathcal{S}} = \arg\min_{h \in H} R^{\ell}_{\mathcal{S}}(h).$$

In this setting, h is chosen from a restricted class of functions H set in advance. Handpicking H can be a difficult task when no prior knowledge about the data is available. Although ERM can perform well in some cases, recall that the empirical error is often a too optimistic estimate for true error. There is always a hypothesis h complex enough to reproduce exactly the labels of the training set, making this framework suffer from overfitting.

In order to avoid overfitting, one should use the most simple hypothesis that explains the data, which is supported by Occam's razor principle. Two main approaches can be envisaged: limiting the class of functions H (through structural risk minimization) or favoring simple hypotheses over more complex ones (through regularized risk minimization). We present both these approaches in the following.

Structural Risk Minimization In the case of *structural risk minimization* (SRM), an infinite number of hypothesis classes $H_1 \subset H_2 \subset \dots$ of increasing sizes and complexities is considered. The hypothesis is chosen to fit the data while staying as simple as possible. This is done using a penalty on the complexity of the hypotheses class:

$$h_{\mathcal{S}} = \arg\min_{h \in H_n, n \in \mathbb{N}} R^{\ell}_{\mathcal{S}}(h) + \text{penalty}(H_n).$$

The penalty is proportional to a measure of complexity of H_n (e.g. the Vapnik-Chervonenkis dimension (Vapnik & Chervonenkis, 1971)). This framework can be difficult to implement, as these measures can be sometimes hard to estimate, even for simple classes of hypotheses.

Regularized Risk Minimization Like SRM, Regularized Risk Minimization (RRM) builds upon limited spaces of hypotheses. Here, we choose one large hypothesis space that will be restricted through a penalty function, also known as a *regularizer*. The regularizer usually takes the form of a norm on h ($\|h\|$). The purpose is to reach a trade-off between best fitting the training data by minimizing the empirical risk, and controlling the complexity of the hypothesis:

$$h_{\mathcal{S}} = \arg\min_{h \in H} R^{\ell}_{\mathcal{S}}(h) + \lambda \|h\| .$$

Choosing a type of regularizer for RRM is a delicate task, as each norm type enforces different characteristics on the model. Moreover, some choices make the problem easier to solve than others due to their good mathematical properties, e.g. convexity and/or smoothness. Table 2.1 presents the most common regularizers used in classification and their characteristics. The choice of the regularizer depends on the properties of the problem. The L_2 norm is easy to optimize and provides control over the values of a vector; the same

Table 2.1: Common regularizers on vectors ($\|\mathbf{x}\|$) and matrices ($\|\mathbf{M}\|$).

Name	Definition	Properties
L_0 norm	$\|\mathbf{x}\|_0 = \sum \mathbb{1}_{x_i \neq 0}$ $\|\mathbf{M}\|_0 = \sum \mathbb{1}_{M_{ij} \neq 0}$	+ Sparsity - Non-convex, non-smooth
L_1 norm	$\|\mathbf{x}\|_1 = \sum \|x_i\|$ $\|\mathbf{M}\|_1 = \sum \|M_{ij}\|$	+ Convex, sparsity - Non-smooth
(Squared) L_2 norm	$\|\mathbf{x}\|_2^2 = \sum x_i^2$	+ Strongly convex, smooth
(Squared) Frobenius norm	$\|\mathbf{M}\|_{\mathcal{F}} = \sum M_{ij}^2$	+ Strongly convex, smooth
$L_{2,1}$ norm	$\|\mathbf{M}\|_{2,1} = \sum_{j=1}^{n} \|\mathbf{M}_{\cdot,j}\|_2$	+ Convex, group sparsity - Non-smooth
Nuclear (trace) norm	$\|\mathbf{M}\|_* = \sum$ singular values	+ Convex, low rank - Non-smooth

is true for its matrix counterpart, the Frobenius norm. However, when the data contains a large number of features and potentially not large enough number of examples, one would consider building a model relying only on a small number of features. The sparsity inducing norms, e.g. the L_1 and $L_{2,1}$ norms, can be used to achieve this purpose. As stated before, using regularization limits the complexity of the model to avoid overfitting and obtain better generalization. We will show in Section 2.7 how controlling the model through regularization can also help obtain theoretical guarantees on how well the model will generalize to unseen data.

2.4 Loss Functions

All the strategies presented earlier for finding a good model are based, at least in part, on minimizing the empirical error under a certain loss function ℓ. Although the zero-one loss $\ell_{0/1}$ seems to be the most adapted for classification, this function is non-convex, non-smooth, and optimizing it directly is known to be NP-hard (Ben-David et al., 2003). In practice, we replace it with a surrogate loss that approximates the real loss and has better computational properties. We remind the reader that a loss function is defined as $\ell : H \times \mathcal{Z} \to \mathbb{R}^+$. We now present the most frequently used margin-based surrogates for binary classification.

- The hinge loss is defined as:

$$\ell_{hinge}(h, z) = [1 - yh(\mathbf{x})]_+ = \max(0, 1 - yh(\mathbf{x})).$$

  It provides a relatively tight and convex upper bound for the zero-one loss, but it is not smooth (i.e. not differentiable) for the margin to be satisfied, that is $yh(\mathbf{x}) = 1$. The hinge loss is closely related to support vector machines (SVMs) (Cortes & Vapnik,

1995a).

- The logistic loss

$$\ell_{log}(h, z) = \ln(1 + e^{-yh(\mathbf{x})}).$$

This function has a similar convergence rate as the hinge loss, being also smooth. Its main disadvantage is that its value is never zero, even for correctly classified points, which only incur a small penalty. This property makes logistic loss sensitive to outliers. The logistic loss is often associated with logistic regression (Knoke & Burke, 1980).

- The exponential loss, defined as:

$$\ell_{exp}(h, z) = e^{-yh(\mathbf{x})},$$

has similar properties to the logistic loss, but its values are closer to zero for correct predictions. The exponential loss is mostly known by being used in boosting (Freund & Schapire, 1995).

Figure 2.1 depicts these surrogate loss functions, along with the zero-one loss. Note that all these surrogates give a loss penalty of 1 for $yh(\mathbf{x}) = 0$.

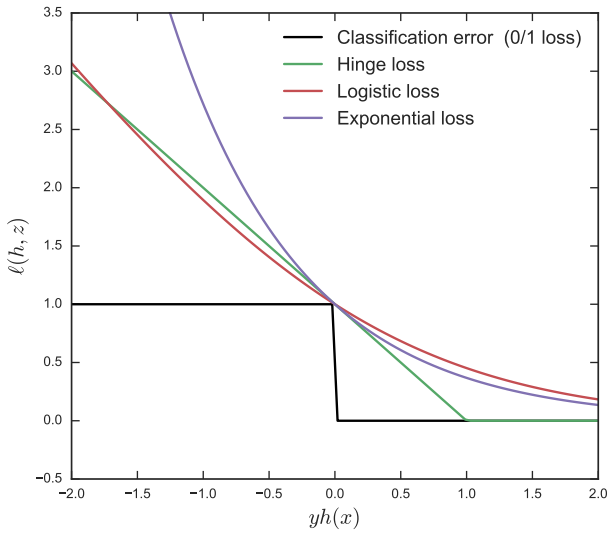


Figure 2.1: Loss functions for binary classification.

2.5 Standard Classification Algorithms

In this section, we present two classic methods for supervised learning. Support vector machines (Section 2.5.1) represent one of the most largely used types of classifiers. The contributions we present in the following chapters are strongly related to this method, as we will see. The k-NN algorithm presented in Section 2.5.2 is one of the simplest, yet effective classification rules. It is also one of the methods that has been most explored in

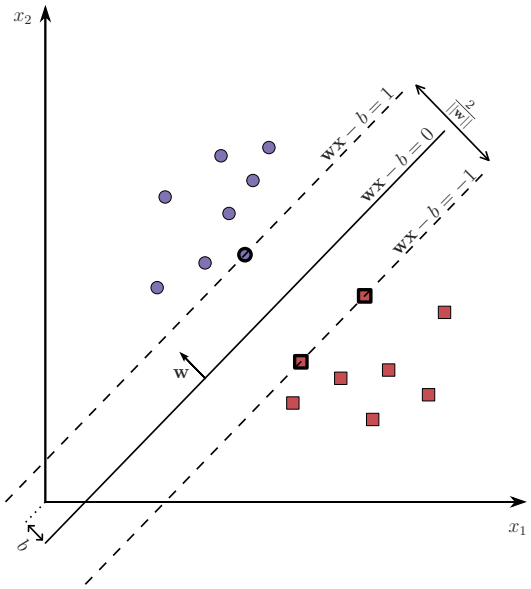


Figure 2.2: Maximum-margin hyperplane and margins for a binary SVM. Support vectors are marked with a black outline.

relation to learning metrics, as we will detail in the review of the state of the art methods for metric learning (Chapter 3).

2.5.1 Support Vector Machines

Support vector machines (SVM) (Vapnik & Chervonenkis, 1964) are supervised methods adapted to classification and regression. We present here their formulation for binary classification. The algorithm computes the best separating hyperplane between the classes, i.e. the one that maximizes their separation. In their initial version, SVMs were only adapted for finding a linear separator, but this setting was extended to nonlinear decision functions using the so-called kernel trick (Boser et al., 1992) and projecting data in high-dimensional or even infinite spaces. For these two settings, the classes need to be linearly separable respectively in the input space and in the feature (projection) space. An important advantage of SVMs is that the equation of the separator only depends on the training points placed exactly on the required margin, also known as support vectors (Figure 2.2). The notion of margin of a separator with respect to a training set is defined to be the minimal distance between a point in the training set and the separator. Intuitively, between two predictors with the same error but different margins, the one with the larger margin is to be preferred, because it has a higher probability of still separating the training set under slight perturbations of the instances. SVMs aim at finding the maximum margin separator. Large margins on the training set translate into better upper bounds on the true risk.

This constraint is relaxed in the soft margin SVM (Cortes & Vapnik, 1995b), which is able to compute a large margin separator while allowing for a small proportion of examples to violate the margin. When the classes are separable, choosing a very small margin in the

soft margin SVM yields the previous hard margin formulation. Computing the classifier amounts to solving the following problem:

$$\min_{\mathbf{w},b} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(\mathbf{w}\mathbf{x}_i + b)]_+ + \lambda \|\mathbf{w}\|_2^2 .$$

The previous formulation learns a linear separator, but, as stated before, SVMs can be used to learn nonlinear classifiers. For this, we suppose that a kernel function K that measures similarity between points is given. We now give the definition of a kernel function.

Definition 2.5 (Mercer kernel)**.** *A pairwise similarity function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a Mercer kernel if it is symmetric and positive semi-definite (PSD), i.e.*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

for all finite sequences $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ and $c_1, \ldots, c_n \in \mathbb{R}$.

These properties of kernel functions are the basis for the kernel trick and dealing with classes that are not linearly separable. Positive semi-definiteness implies the existence of a Hilbert space $\mathbb{H}$ to which the data can be projected in order to obtain linear separation. The projection is done through a nonlinear function, which can be unknown or implicit.

Theorem 2.6 (Mercer theorem (Mercer, 1909))**.** *A kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is symmetric and positive semi-definite if and only if there exists a projection function ϕ from the instance space $\mathcal{X}$ to a Hilbert space $\mathbb{H}$ such that $K(\cdot, \cdot)$ can be written as a proper inner product:*

$$K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle, \quad \forall \mathbf{x}, \mathbf{x}'.$$

We provide some examples of kernels when discussing metrics for feature vectors (see Section 3.2.1). Given a kernel K, the equation of the separating hyperplane of an SVM is:

$$\mathbf{w} = \sum_{i=1}^{n} c_i y_i \phi(\mathbf{x}_i).$$

The advantage of the kernel trick is that the mapping function ϕ can be unknown and is not necessary for solving the problem. Computing the classifier can be done by learning the values of the coefficients c_i based on the kernel K:

$$\begin{aligned} \max_{c_i, i \in \{1,\ldots,n\}} & \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to } & \sum_{i=1}^{n} c_i y_i = 0 \\ & 0 \leq c_i \leq \frac{1}{2n\lambda}, \forall i \in \{1, \ldots, n\}. \end{aligned}$$

Finally, new points $\mathbf{x}$ can be classified to the positive (+1) or negative (-1) class using the

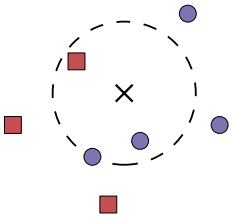


Figure 2.3: Example of 3-NN classifier using the Euclidean distance. Here, the new instance (cross) will be assigned to the circles class.

Algorithm 2.1 k-nearest neighbors algorithm

Input Training sample $\mathcal{S}$, neighborhood size k, distance $d(\cdot,\cdot)$, decision rule function $c(\cdot)$, example to classify x
Output $y =$ the label of x
$N \leftarrow Neighbors(x, \mathcal{S}, k, d)$ ▷ Determine the k-NN of x from $\mathcal{S}$ under distance d
$y \leftarrow c(N)$ ▷ Combine the labels of the k neighbors
return y

following prediction rule:

$$y = \operatorname{sgn}\left(\sum_{i=1}^{n} c_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right).$$

2.5.2 Nearest Neighbor Classifier

The k nearest neighbors (k-NN) algorithm (Cover & Hart, 1967) is one of the most used classifiers, due to its simple classification rule. In contrast to SVM, the model is only approximated locally, without computing its overall explicit form. k-NN does not make any assumptions about the underlying distribution of the data, thus being non parametric. This is a lazy algorithm, as the training phase only stores the data, computation being deferred until the prediction phase. Upon the arrival of a new, unlabeled sample, its label is predicted as the most frequent class among the k training samples nearest to the point w.r.t. some distance function (see Algorithm 2.1 and Figure 2.3).

As k-NN uses a rule of majority voting, an important issue is breaking ties among the voters when two or more classes are majoritary. In a binary setting, choosing an odd value for k is enough to ensure that ties are not possible. In multiclass setting, a trivial heuristic is to break ties equally by randomly choosing one of the majority classes. Another solution is Distance Weighted k-NN (Dudani, 1976), which weighs the neighbors' votes proportionally to the inverse of their distance to the example.

Theoretically speaking, the higher k, the better the results of k-NN. However, this only holds when the number of examples n is infinite. In finite situations, the best choice for the size of the neighborhood k depends on the data. In general, larger values reduce the effect of noise but also smooth the decision boundary, making it harder to classify borderline examples.

Table 2.2: Confusion matrix for classification error.

		Predicted value	
		Positive	Negative
True label	Positive	True positive (tp)	False negative (fn)
	Negative	False positive (fp)	True negative (tn)

The right value is usually selected through heuristics of hyperparameter optimization (see Section 2.6).

A significant drawback of k-NN is the necessity to store the training set during prediction. As in practice not all the training points are needed for accurate classification, data reduction can be performed by keeping in the training set only the examples that help correctly classify all the others. This type of politic is used in variants of k-NN like condensed nearest neighbor (CNN) (Hart, 1968) and reduced nearest neighbor (RNN) (Gates, 1972).

From a theoretical point of view, the classic k-NN algorithm has the property of its risk converging to the Bayes error when the number of examples n grows, for all values of k:

$$R_P^{Bayes} = \mathbb{E}_{(x,y)\sim P}\left(1 - \max_{y\in\mathcal{X}} \Pr(y|x)\right) \Pr(x).$$

In the binary case, when the number of classes is limited to two, the following inequality holds:

$$R_P^{Bayes} \leq R_P^{k\text{-NN}} \leq 2R_P^{Bayes}.$$

The performance of the algorithm is sensitive to the metric through which the neighbors are computed. For feature vectors, the most common choice of metric is the Euclidean distance. A good strategy for improving k-NN performance is to learn a custom distance function from the data through metric learning. As we will see in Chapter 3, an important number of methods from this field has been dedicated to learning distances that minimize some form of empirical risk for k-NN classification.

2.6 Algorithms Evaluation

The performance of supervised machine learning algorithms can be compared through multiple measures. We provide here a discussion over these measures, data partitioning and parameter selection techniques for classification.

We start by illustrating all possible outcomes of the prediction for binary classification with respect to the actual target values in Table 2.2. The terms positive and negative refer to the classifier's prediction, while the terms true and false refer to whether that prediction corresponds to the instance label. The scores that are often computed from such a matrix are the following.

- Precision is referred to as positive predictive value:

$$\text{precision} = \frac{\text{tn}}{\text{tn} + \text{fp}}.$$

- Recall is referred to as the true positive rate or sensitivity:

$$\text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}}.$$

- The F-score (or F_1 score) is a measure that combines precision and recall as a harmonic mean:

$$\text{F-score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

- Accuracy is a statistical measure of how well a binary classification model correctly identifies or excludes a condition. That is, the accuracy is the proportion of true results, both positives and negatives, among the total number of cases examined:

$$\text{accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}.$$

- Conversely, we can compute the error rate as:

$$\text{error} = 1 - \text{accuracy}\,.$$

Computing the error rate is the same as evaluating the zero-one loss over the whole sample. Throughout this dissertation, we will mostly rely on the accuracy and error rates for classification algorithms.

As we are interested in the performance of a model on unseen data, evaluating the predictive quality of a model should not be done on the same sample that has been used for learning. An additional reason is that the error on the training set is usually a too optimistic estimator of the model performance on unseen data.

To simulate new data from a finite sample, but still have access to the correct output, a common strategy is to take all available labeled data and randomly split it into training and test subsets. The training sample is then used by the algorithm to learn the model, while the test sample serves for evaluating the model and is unaccessible during training. Other than the parameters learned by a model, the vast majority of methods depend on additional values (*hyperparameters*) that cannot be learned from the training set. They often leverage a trade-off, such as regularization parameters. These hyperparameters need to be tuned, and an additional independent sample is necessary to choose the best value. This validation sample is also set aside from the labeled sample, more specifically from the training set, as hyperparameter tuning is part of the learning process.

The traditional way for hyperparameter optimization is *grid search*, which exhaustively evaluates all possible combinations of parameters. For each setting, the model is estimated on a training set, then evaluated on the validation data. The values that performed the best are retained, and the model is trained once more, this time on the training set augmented

with the validation set. The final model performance is computed on the test set.

Since the parameter space is often continuous, the value ranges and discretization steps have to be set manually for each parameter. When the number of hyperparameters and values to be explored is large, the size of the state space increases exponentially, making exhaustive grid search prohibitively expensive. Several alternatives to exhaustive search have been proposed. In particular, randomly sampling parameters a fixed number of times is an effective alternative in high-dimensional spaces (Bergstra & Bengio, 2012). Both full and randomized grid searches are completely parallelizable.

By setting aside data for validation and testing, the size of the training set is diminished. Generally, the larger the training data the better the classifier, but also the larger the test data the more accurate the error estimate. In order to better exploit the data, *cross-validation* allows to perform multiple rounds of evaluation on different splits of the data. Multiple heuristics for these splits exist:

- Multiple random splits (subsampling or repeated holdout method).
- K-fold cross-validation: the original sample is randomly partitioned into K mutually exclusive subsets (the "folds") of approximately equal size. Two folds are retained respectively for validation and testing, while the other $K - 2$ folds are used for training.
- Leave-one-out cross-validation: on each run, one point is excluded from the training sample; it is instead used for testing. The learning process is performed once for each sample.

The price to pay for the previous heuristics comes as an increased time complexity.

2.7 Generalization Guarantees

In the previous sections, we have presented general frameworks for minimizing different versions of the empirical risk over a given sample. However, the objective of learning is to obtain a model with good generalization properties for unseen data. Generalization bounds allow to explicitly relate true risk to the empirical error of a hypothesis and to measure to which extent the empirical risk deviates from the true value over a sample. When the difference between the two measures is small, minimizing the (penalized) empirical error will have a high probability of yielding a model with low true error. This type of probabilistic bound is referred to as a Probably Approximately Correct (PAC) bound.

Generalization bounds measure the deviation of a random variable from its expected value, making concentration inequalities highly useful mathematical tools for deriving them. The most commonly used in practice are Hoeffding's inequality in the case of sums of variables and McDiarmid's inequality for bounded functions. We recall them in Appendix A.

We start by formally defining PAC learning, then move on to presenting three theoretical frameworks for deriving PAC generalization bounds: uniform convergence, uniform stability

and algorithmic robustness. These frameworks will be used in Chapters 4 and 5 to derive generalization guarantees for the proposed learning methods.

2.7.1 PAC Learning

The idea of Probably Approximately Correct (PAC) learning (Valiant, 1984) is that learning an unknown target concept should be considered successful only if a hypothesis good for approximating it can be found with high probability. The goal is thus that, with high probability (probably), the selected model function will have low generalization error (approximately correct).

The original framework is designed for learning the output of boolean functions. The instance space is assumed to be the set of all possible assignments of n Boolean variables $\mathcal{X} = \{0,1\}^n$, and the probability of each instance is given by the distribution P over $\mathcal{X}$. In this context, a *concept* c is a subset of $\mathcal{X}$. A *concept class* C is a set of concepts over $\mathcal{X}$. We can define PAC learnability as follows.

Definition 2.7 (PAC-learnable)**.** *Given $0 < \delta, \epsilon < 1$, a concept class C is learnable by a polynomial time algorithm A if, for any distribution P of samples and any concept $c \in C$, there exists a polynomial $p(\cdot,\cdot,\cdot)$ such that A will produce with probability at least $1-\delta$ a hypothesis $h \in C$ whose error is less than or equal to ϵ when given at least $p(n, 1/\delta, 1/\epsilon)$ independent random examples.*

The PAC framework was the first one to convey the notion of efficiency in learning in a similar way to complexity theory. The learning process must take place in polynomial time of the sample size and must use only a reasonable amount of examples, while providing a good approximation of the target concept. The smallest polynomial p for which learning C is possible represents the *sample complexity* of the algorithm A. The model is worst case, as it defines a unique bound for all target concepts and all distributions over the instance space.

2.7.2 Uniform Convergence

Uniform convergence analysis (Vapnik & Chervonenkis, 1971) studies the concentration of empirical quantities towards their expected value. It is one of the most important tools for deriving PAC generalization bounds. These guarantees hold with high probability (usually $1-\delta, \delta > 0$) for a given hypothesis class and depend on the size of the training sample. Through the law of large numbers, large samples provide higher confidence, while small samples or too large hypothesis classes come with small confidence and an important risk of overfitting. The classic convergence rate for this type of bounds is $\mathcal{O}(1/\sqrt{n})$ in the size of the sample.

We have previously given the intuition of hypothesis complexity and its link to predictor performance. In practice, different measures exist for this complexity. In the case of finite hypothesis spaces, we get the following PAC bound:

Theorem 2.8 (Uniform convergence bound for finite hypothesis spaces)**.** *Let $\mathcal{S}$ be a sample of size n drawn i.i.d. from a distribution P, H a finite hypothesis space of cardinality $|H|$ and $\delta > 0$. Then, with probability $1 - \delta$ over the sample $\mathcal{S}$, we have:*

$$R_P^\ell(h) \leq R_\mathcal{S}^\ell(h) + \sqrt{\frac{\ln |H| + \ln (1/\delta)}{2n}}.$$

In the case of a continuous hypothesis space, $|H|$ is no longer a finite value, thus the previous bound is no longer useful. Instead, we need measures that are capable of evaluating the complexity of a hypothesis space without considering its cardinality. These include the VC dimension (Vapnik & Chervonenkis, 1971), the fat-shattering dimension (Kearns & Schapire, 1994), the maximum discrepancy (Bartlett et al., 2002), the Rademacher complexity (Bartlett & Mendelson, 2003; Koltchinskii, 2001) and the Gaussian complexity (Bartlett & Mendelson, 2003).

2.7.2.1 Vapnik-Chervonenkis Dimension

The Vapnik-Chervonenkis (VC) dimension (Vapnik & Chervonenkis, 1971) is one of the most important results in statistical learning theory. It has allowed to extent the learning theory from finite spaces to infinite ones, and is based on the observation that what matters is not the cardinality of H, but rather its expressive power.

Let us say we have a sample $\mathcal{S}$ containing n points. Intuitively, these n points can be labeled in 2^n ways as positive and negative, therefore defining 2^n labeling combinations. If for any of these problems, we can find a hypothesis $h \in H$ that separates the positive examples from the negative, then we say H *shatters* n points, i.e. any learning problem definable by n examples can be learned with no error by a hypothesis drawn from H. The maximum number of points that can be shattered by H is called the Vapnik-Chervonenkis (VC) dimension of H. We define this notion formally in the following way.

Definition 2.9 (VC dimension (Vapnik & Chervonenkis, 1971))**.** *The VC dimension of a set of indicator functions H is the maximum number n of vectors $x_1, \ldots, x_n$ that can be separated into two classes in all 2^n possible ways using functions from H. If for any n there exists a set of n vectors that can be shattered by the set H, then the VC dimension is equal to infinity.*

To illustrate this concept, consider the space of all linear classifiers in the plane (Figure 2.4). The VC dimension of linear functions in the plane is 3 but not 4, since no four points can be shattered by a set of lines. In general, a linear function in an n dimensional space has VC dimension $n + 1$, which is also the number of free parameters.

Definition 2.9 has been extended to real-valued functions in (Vapnik, 1979) and can be used to obtain generalization bounds of the following form.

Theorem 2.10 (Uniform convergence bound with VC dimension)**.** *Let $\mathcal{S}$ be a sample of size n drawn i.i.d. from a distribution $\mathcal{P}$, H a hypothesis space of finite VC dimension*

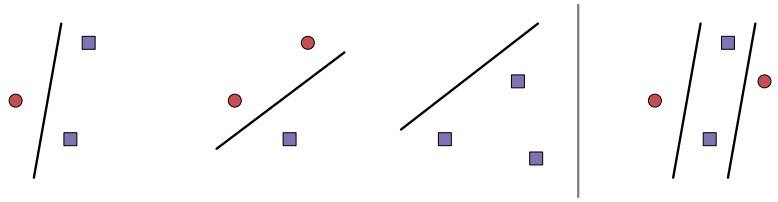


Figure 2.4: VC dimension of a linear classifier in the plane. A line can shatter three points with all possible labeling combinations (on the left, three out of eight combinations), but it cannot shatter all sets of four points (on the right). The VC dimension of a linear separator in the plane is 3.

$\mathrm{VC}(H)$ *and* $\delta > 0$. *Then, with probability* $1 - \delta$ *over the sample* $\mathcal{S}$*, we have:*

$$R_P^\ell(h) \leq R_\mathcal{S}^\ell(h) + \sqrt{\frac{\mathrm{VC}(H)\left(\ln \frac{2n}{\mathrm{VC}(h)} + 1\right) + \ln(4/\delta)}{n}}.$$

It is worth noticing that a finite VC dimension for a hypothesis class H is equivalent to H being PAC-learnable under some regularity conditions. The VC dimension is a pessimistic measure, as it is distribution-free and evaluates the capacity of a class of functions in the worst case. On one hand, this is sometimes an advantage because it guarantees the bounds to hold for any data distribution. On the other hand, the bounds might not be tight for certain data distributions. Moreover, it does not take into account the way the algorithm learned the hypothesis h. In practice, the use of the VC dimension is limited, as some of classic machine learning algorithms have infinite VC dimensions (k-NN classifier, SVMs with certain kernels) and thus cannot be analyzed through this measure. We now turn to the Rademacher complexity, which is a sample-dependent measure that can provide a better insight into certain classes of functions than the VC dimension.

2.7.2.2 Rademacher Complexity

The uniform convergence bound based on the VC dimension is distribution independent. In contrast, the Rademacher complexity is a more recent measure of complexity that allows to factor in the data distribution, or its empirical estimate. The method of Rademacher symmetrization has been thoroughly exploited in the empirical processes theory (Gine & Zinn, 1984; Koltchinskii, 1981; Pollard, 1982), but its use in statistical inference has been limited until more recent years (Koltchinskii, 2001).

Definition 2.11 (Empirical Rademacher complexity)**.** *Let* $\mathcal{S}$ *be a sample of size* n *drawn i.i.d. from a distribution* P *and* H *be a class of uniformly bounded functions. For every integer* n*, the empirical Rademacher complexity over* H *is*

$$\hat{\mathfrak{R}}_n(H) = \mathbb{E}_\sigma \left[\sup_{h \in H} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(z_i) \right],$$

where $\{\sigma_i : i \in \{1, \ldots, n\}\}$ *are independent Rademacher random variables, that is,* $\Pr(\sigma_i =$

$1) = \Pr(\sigma_i = -1) = \frac{1}{2}$.

Definition 2.12 (Rademacher complexity)**.** *The Rademacher complexity of H is defined as*

$$\mathfrak{R}_n(H) = \mathbb{E}_{\mathcal{S}}\hat{\mathfrak{R}}_n(H).$$

Intuitively, for a given $\mathcal{S}$ and Rademacher vector σ, the supremum measures the maximum correlation between $h(z_i)$ and σ_i over all $h \in H$, that is the correlation of a learned hypothesis to a randomly relabeled sample. $\mathfrak{R}_n$ can be viewed as a measure of capacity of H to separate classes. As shown in (Koltchinskii, 2001), if the value of $\mathfrak{R}_n$ is large, the class of hypothesis H will separate positive from negative examples even if the labels are assigned randomly. This is an indication that the hypothesis class is too large: a good choice for H would separate well the examples only when labels are assigned correctly. We now show a uniform convergence result for any class of bounded real-valued functions.

Theorem 2.13 (Uniform convergence bound using Rademacher complexity)**.** *With probability at least $1-\delta$ over $\mathcal{S}$, every function h in H satisfies*

$$R_P^{\ell}(h) \leq R_{\mathcal{S}}^{\ell}(h) + 2\mathfrak{R}_n(H) + \sqrt{\frac{\ln(1/\delta)}{n}}.$$

Moreover, with probability at least $1-\delta$ over $\mathcal{S}$, for every function h in H,

$$R_P^{\ell}(h) \leq R_{\mathcal{S}}^{\ell}(h) + 2\hat{\mathfrak{R}}_n(H) + 3\sqrt{\frac{\ln(2/\delta)}{n}}.$$

Even though uniform convergence bounds do not take into account parameters from the learning algorithm, the Rademacher complexity term in the previous theorem can sometimes implicitly consider the impact of the regularization term. The second bound shows that the true risk can be bounded with respect to the empirical risk and the empirical Rademacher complexity, computed based on only one sample and one random vector σ. Note that we will be using Rademacher complexity analysis in Chapter 4.

Independently of the choice of measure of complexity, uniform convergence analysis yields bounds based on the size of the training sample and the class of hypothesis selected. The main drawback of this characterization is that the bound does not take into account the learning algorithm, i.e. how the hypothesis $h_{\mathcal{S}}$ is selected. We will now present two analytical frameworks that allow to derive generalization guarantees for a hypothesis $h_{\mathcal{S}}$ based on the properties of the learning algorithm in a setting of regularized risk minimization.

2.7.3 Uniform Stability

Algorithmic stability (Bousquet & Elisseeff, 2001, 2002) is a framework based on sensitivity analysis, determining how much a variation in the input of a system can influence its output. The motivation in learning for such an analysis is to design robust algorithm that will not be affected by the sampling mechanisms providing the training set. From the different notions of stability, we will focus on uniform stability, the strongest notion from this family.

Recall that $\mathcal{X}$ and $\mathcal{Y} \subset \mathbb{R}$ are respectively the input and the output space. Consider a labeled training sample $\mathcal{S} = \{z_i\}_{i=1}^n$ of size n in $\mathcal{X} \times \mathcal{Y}$ drawn i.i.d. from an unknown distribution P. We will denote by $\mathcal{S}^i$ the training set obtained from $\mathcal{S}$ by replacing the ith example with a new independent one coming from the same distribution. In this setting, we can define uniform stability for a learning algorithm.

Definition 2.14 (Uniform stability (Bousquet & Elisseeff, 2002))**.** *Given a loss function ℓ, a learning algorithm has a uniform stability in $\frac{\kappa}{m}$ with respect to ℓ, with $\kappa \geq 0$ constant, if for all i,*

$$\forall \mathcal{S}, |\mathcal{S}| = n, \sup_z |\ell(h_{\mathcal{S}}, z) - \ell(h_{\mathcal{S}_i}, z)| \leq \frac{\kappa}{n}.$$

Uniform stability ensures only small variations of the learned hypothesis are possible under minor perturbations in the training set. This property enables us to derive a generalization bound on the true error of an algorithm.

Theorem 2.15 (Uniform stability bound)**.** *Let $\mathcal{S}$ be a sample of size n drawn i.i.d. from a distribution P, ℓ a loss function upper-bounded by a constant B, and $\delta > 0$. For any algorithm A with uniform stability of κ/n with respect to ℓ, with probability $1 - \delta$ over the random sample $\mathcal{S}$, we have:*

$$R_P^\ell(h_{\mathcal{S}}) \leq R_{\mathcal{S}}^\ell(h_{\mathcal{S}}) + \frac{\kappa}{n} + (2\kappa + B)\sqrt{\frac{\ln(1/\delta)}{2n}},$$

for a hypothesis $h_{\mathcal{S}}$ learned by the algorithm A from sample $\mathcal{S}$.

The bounds obtained through uniform stability are relatively tight, as a consequence of taking into account the learning algorithm (more precisely, the loss function, the regularizer and the regularization parameter have an impact on κ). In contrast to uniform convergence, stability does not make any assumptions about the class of hypothesis. Notice that the two frameworks obtain the same convergence rate in the sample size. This framework can be applied to a large number of algorithms (Bousquet & Elisseeff, 2002), including k-NN classification, soft margin SVM classification and least squares regression. Its major drawback is that it can only be used with strongly convex regularization functions. (Kearns & Ron, 1997) have shown that a finite VC dimension implies stability, in the sense that using the stability as a complexity measure does not yield worse bounds that the VC dimension.

2.7.4 Algorithmic Robustness

Algorithmic robustness (Xu & Mannor, 2010, 2012) characterizes the capability of an algorithm to perform similarly on close training and test instances. While uniform stability covers the variations coming from the sample selection heuristic, robustness takes into account the other factor that justifies the randomness in learning, which is sample noise. For a robust algorithm, if a training sample and a test example are close to each other, then their associated loss values must also be close. The notion of closeness is based on a partitioning of the instance space $\mathcal{Z}$ based on covering numbers (Kolmogorov & Tikhomirov, 1961).

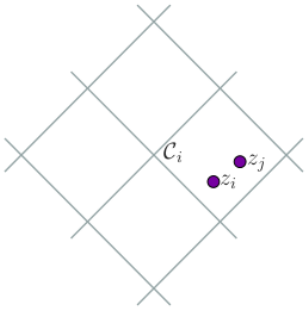


Figure 2.5: Robustness using an L_1 norm cover. The difference in loss between z and z' must be bounded.

Definition 2.16 (Covering number)**.** *For a metric space (S, ρ) and $T \subset S$, we say that $\hat{T} \subset S$ is an ϵ-cover of T, if $\forall t \in T, \exists \hat{t} \in \hat{T}$ such that $\rho(t, \hat{t}) \leq \epsilon$. The covering number of T is:*

$$\mathcal{N}(\epsilon, T, \rho) = \min\{|\hat{T}| : \hat{T} \text{ is an } \epsilon\text{-cover of } T\}.$$

In other words, the covering number is the number of spherical balls of a given size ϵ needed to completely cover a given space, with possible overlaps. The partition is used to determine closeness: two examples are close if they belong to the same region, implying that the norm between them is smaller than a fixed quantity ρ (Figure 2.5). Formally, the robustness is defined as follows.

Definition 2.17 (Algorithmic Robustness (Xu & Mannor, 2010, 2012))**.** *Algorithm A is $(M, \epsilon(\cdot))$-robust, for $M \in \mathbb{N}$ and $\epsilon(\cdot) : \mathcal{Z}^n \to \mathbb{R}$, if $\mathcal{Z}$ can be partitioned into M disjoint sets, denoted by $\{C_i\}_{i=1}^M$, such that the following holds for all $\mathcal{S} \in \mathcal{Z}^n$:*

$$\forall \mathbf{z} = (\mathbf{x}, l(\mathbf{x})) \in \mathcal{S}, \forall \mathbf{z}' = (\mathbf{x}', l(\mathbf{x}')) \in \mathcal{Z}, \forall i \in [M]:$$
$$\text{if } \mathbf{z}, \mathbf{z}' \in C_i, \text{ then } |\ell(A, \mathbf{z}) - \ell(A, \mathbf{z}')| \leq \epsilon(\mathcal{S}).$$

Roughly speaking, an algorithm is robust if for any test example $\mathbf{z}'$ falling in the same subset as a training example $\mathbf{z}$, the gap between the losses associated with $\mathbf{z}$ and $\mathbf{z}'$ is bounded. This notion is a desired property of a learning algorithm, as it implies a lack of sensitivity to small perturbations in the data in the same sense as robust optimization. (Xu & Mannor, 2010, 2012) have shown that robust algorithms generalize well in the following theorem.

Theorem 2.18 (Algorithmic robustness bound)**.** *Let ℓ be a loss function bounded by a constant B, and $\delta > 0$. If an algorithm A is $(M, \epsilon(\cdot))$-robust, then, with probability $1 - \delta$, we have:*

$$R_P^\ell(h_\mathcal{S}) \leq R_\mathcal{S}^\ell(h_\mathcal{S}) + \epsilon(\mathcal{S}) + B\sqrt{\frac{2M \ln 2 + 2\ln(1/\delta)}{n}},$$

where $h_\mathcal{S}$ is the hypothesis learned by A from the sample $\mathcal{S}$.

Note that in the previous bound there is a trade-off between M and $\epsilon(\mathcal{S})$: the smaller the parts in the partition, the more of them there will be. Algorithmic robustness generalization bounds are usually not tight, due to the dependence on potentially large covering numbers.

Nevertheless, one important advantage of this framework is the capacity to integrate a large span of regularizers, including sparsity inducing norms.

2.8 Conclusion

In this chapter, we presented the setting for supervised learning, as well as the main frameworks for deriving generalization guarantees for learning algorithms. Based on these notions, Chapter 3 explains the impact of metrics on algorithm performance, along with presenting state of the methods in metric learning.

Chapter 3

Review of Metric Learning

Chapter abstract

In this chapter, we review the existing methods for supervised and semi-supervised metric learning. We begin by presenting an overview of some standard metrics. Our literature study covers supervised metric learning methods for feature vectors and time series. After discussing these aspects, we are able to conclude over the advantages and limitations of the state of the art methods, which represent the principal motivations for our contributions.

3.1 Introduction

The performance of most types of machine learning algorithms strongly depends on the representation of the data and the metrics used to compare instances. However, selecting an adapted metric or representation can prove to be a difficult task, and the choice should take into account multiple aspects, such as the field of application, the type of data, as well as the task to be accomplished. This is the reason for the important attention received by the topic of representation learning, which includes subjects like metric learning (Bellet et al., 2015), kernel learning (Sonnenburg et al., 2006), feature learning (Contardo et al., 2016), etc. Examples of methods relying on metrics and the representation of the data include learning algorithms with different settings:

- k-nearest neighbors (k-NN) classification (Cover & Hart, 1967), where the notion of neighborhood is defined with respect to a distance function.
- Kernel methods (Schölkopf & Smola, 2002; Shawe-Taylor & Cristianini, 2004), where the data is projected in a new feature space through the use of a kernel function.
- K-means clustering (Lloyd, 2006), which creates clusters of points in a way that minimizes within-class variance under a distance function.
- Information retrieval (Frakes & Baeza-Yates, 1992), where a similarity function is used to find the documents which are closest to a given query.
- Data visualization and dimensionality reduction (Post et al., 2002), where a metric is often used to change the representation of the data.

The rest of this chapter is mainly split between two major parts. Section 3.2 establishes definitions and notations, before addressing the topic of standard metrics for features vectors and time series. Section 3.3 formally introduces metric learning and details the most relevant methods from this field dedicated to feature vectors and time series. This section is mostly dedicated to supervised and semi-supervised settings. Section 3.4 concludes this chapter by a discussion on the overall trend of state of the art methods in metric learning and their current limitations. This allows us to fully motivate the contributions that we propose in the following chapters.

3.2 Metrics

In this section, we provide an extended study of the most well-known measures for comparing vectors of features and time series. We start by formally defining distance and similarity functions.

Definition 3.1 (Distance function)**.** *A distance is a pairwise function* $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ *which satisfies the conditions:*

- $d(x, y) \geq 0$ *(non-negativity),*
- $d(x, y) = 0 \iff x = y$ *(identity of indiscernibles),*
- $d(x, y) = d(y, x)$ *(symmetry),*
- $d(x, z) \leq d(x, y) + d(y, z)$ *(subadditivity or triangle inequality).*

The first two conditions define a positive definite function. By relaxing the second condition, d becomes a pseudo-metric. The property of triangle inequality has already been vastly exploited to speed-up certain learning algorithms such as k-NN and K-means.

Definition 3.2 (Similarity function)**.** *A similarity function is a pairwise function* $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$*. K is called symmetric if* $\forall x, x' \in \mathcal{X}, K(x, x') = K(x', x)$*.*

Similarity functions can be any pairwise functions assigning scores to pairs of points. Normally, they act in some sense as the inverse of distances: their values are high for similar (close) examples and low for distant ones, while dissimilarity functions behave like normalized distance functions. A special case of similarity functions are kernels, as presented in the previous chapter (Definition 2.5), which enforce the property of positive semi-definiteness.

3.2.1 Metrics for Feature Vectors

Minkowski distances Minkowski distances are a family of distances induced by the L_p norms, defined for $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ and $p \geq 1$ as

$$d_p(\mathbf{x}, \mathbf{x}') = \left\|\mathbf{x} - \mathbf{x}'\right\|_p = \left(\sum_{i=1}^{d} |x_i - x'_i|^p\right)^{1/p}. \tag{3.1}$$

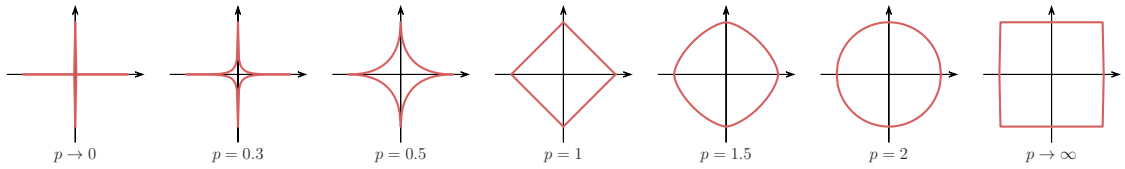


Figure 3.1: Unit circles for various values of p in Minkowski distances.

When $p < 1$, the previous measure is not a proper distance anymore, since it violates triangle inequality. Minkowski distance is typically used with p being 1 or 2.

When $p = 1$, we recover the Manhattan distance:

$$d_1(\mathbf{x}, \mathbf{x}') = \left\|\mathbf{x} - \mathbf{x}'\right\|_1 = \sum_{i=1}^{d} |x_i - x'_i|.$$

Similarly, for $p = 2$, we get the Euclidean distance:

$$d_2(\mathbf{x}, \mathbf{x}') = \left\|\mathbf{x} - \mathbf{x}'\right\|_2 = \left(\sum_{i=1}^{d} (x_i - x'_i)^2\right)^{1/2} = \sqrt{(\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}')}.$$

The Euclidean distance has the additional properties of being translation and rotation invariant, while Manhattan distance is only translation invariant. Note that Euclidean distance assumes all variables are independent and that variance across all dimensions is one, a scenario that is hardly achieved in real world. The limiting case of p reaching infinity defines the Chebyshev distance:

$$d_\infty(\mathbf{x}, \mathbf{x}') = \left\|\mathbf{x} - \mathbf{x}'\right\|_\infty = \lim_{p \to \infty} \left(\sum_{i=1}^{d} |x_i - x'_i|^p\right)^{1/p} = \max_i |x_i - x'_i|.$$

Figure 3.1 shows unit circles with various values of p.

Mahalanobis distance The Mahalanobis distance (Mahalanobis, 1936) was originally defined as a measure of closeness between a point and a distribution. Given a vector $\mathbf{x} \in \mathbb{R}^d$ from a sample $\mathcal{S}$ of mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix Σ^{-1}, the Mahalanobis distance of $\mathbf{x}$ is:

$$d_{\Sigma^{-1}}(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})}.$$

In a similar way, we can define the Mahalanobis distance between a pair of vectors $\mathbf{x}$ and $\mathbf{x}'$ of the same distribution with the covariance matrix Σ^{-1}:

$$d_{\Sigma^{-1}}(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}')}.$$

Currently, the term Mahalanobis distance often refers to its following variation, known also as the generalized ellipsoid distance (Ishikawa et al., 1998):

$$d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')},$$

where $\mathbf{M} \in \mathbb{S}_+^d$. $\mathbb{S}_+^d$ denotes the cone of symmetric PSD $d \times d$ real-values matrices. This constraint on $\mathbf{M}$ makes $d_{\mathbf{M}}$ a pseudo-metric. Setting $\mathbf{M}$ to the identity matrix yields the standard Euclidean distance. An intuition about the transformation applied through $\mathbf{M}$ can be found using Cholesky decomposition. Rewriting $\mathbf{M}$ as $\mathbf{L}^T\mathbf{L}$, where $\mathbf{L} \in \mathbb{R}^{r \times d}$ and r is the rank of $\mathbf{M}$, gives:

$$\begin{aligned} d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') &= \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')} \\ &= \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{L}^T \mathbf{L} (\mathbf{x} - \mathbf{x}')} \\ &= \sqrt{(\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{x}')^T (\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{x}')}. \end{aligned}$$

This decomposition shows that the Mahalanobis distance implies computing the Euclidean distance after a linear projection of the data using the matrix $\mathbf{L}$. When $r < d$, $\mathbf{M}$ is low-rank, and the projection through $\mathbf{L}$ allows for a more compact representation of the data and cheaper computations. Because of its properties and intuitive interpretability, the Mahalanobis distance has attracted a considerable amount of interest in the metric learning community, as we will see in Section 3.3.

Cosine similarity The cosine similarity is a measure of similarity between two vectors based on the cosine of the angle between them:

$$\cos(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^T \mathbf{x}'}{\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2}.$$

Its values range from -1, signifying complete dissimilarity, to 1, meaning exactly the same. When its value is 0, the cosine similarity indicates orthogonality (decorrelation). Figure 3.2 compares the behavior of the cosine similarity against the one of the Euclidean distance. Cosine similarity is widely used in text mining, web mining and information retrieval (Bao et al., 2003; Grabowski & Szałas, 2000; Hust, 2004), where documents are represented as vectors of words indicating term frequencies.

Bilinear similarity The bilinear similarity is a generalization of the cosine similarity parameterized by the matrix $\mathbf{M}$:

$$K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{M} \mathbf{x}',$$

where $\mathbf{M}$ is not required to be PSD. If the data is normalized, setting $\mathbf{M}$ to the identity matrix yields the cosine similarity. The advantage of the bilinear similarity w.r.t. the

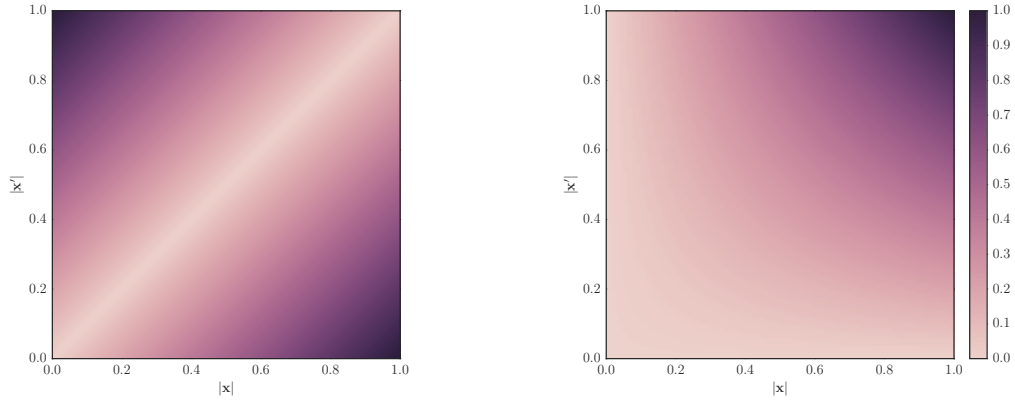


Figure 3.2: Metric behavior for the Euclidean distance (left) and the cosine similarity (right).

cosine similarity, Minkowski and Mahalanobis distances resides in the fact that choosing a non-square matrix $\mathbf{M}$ enables the computation of scores between vectors of different dimensions. The contributions introduced in this dissertation propose novel methods for determining the matrix $\mathbf{M}$ for this similarity.

Linear kernel The simplest linear kernel is the dot product between instances in the original space $\mathcal{X}$:

$$K_{lin}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \cdot \mathbf{x}'.$$

This kernel is equivalent to the cosine similarity without normalization or the bilinear similarity when $\mathbf{M} = \mathbf{I}$.

Polynomial kernel

$$K_{pol}(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^p,$$

where $p \in \mathbb{N}$ is the degree of the polynomial. K_{pol} projects the data into a nonlinear space of all monomials of degree up to p.

Gaussian kernel The Gaussian kernel, also known as the radial basis function (RBF), is expressed as:

$$K_{RBF}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2}\right),$$

where $\sigma^2 > 0$ is a variance (width) parameter. This kernel is a good example of a nonlinear implicit projection into an infinite-dimensional space.

3.2.2 Metrics for Time Series

The presence of time series in numerous fields of application makes them the object of considerable research effort for their classification or prediction, with impact in fields like

speech and writing recognition (Itakura, 1975), energy consumption (Dachraoui et al., 2015), object identification and tracking (Papadimitriou et al., 2005), bioinformatics (Aach & Church, 2001), patient care (Tormene et al., 2009), and many more. To solve such tasks, one is inherently brought to compare time series by pairs, in order to determine their closeness or common patterns. In this section, we present the main types of approaches for evaluating the distance or similarity between time series.

The Euclidean distance for two univariate time series X and X' treats time moments as different features:

$$d_2(X, X') = \left(\sum_{t=1}^{t_X} (X_i - X'_i)^2 \right)^{1/2}.$$

Short time series (STS) distance (Möller-Levet et al., 2003) is designed for unevenly sampled time series, by explicitly modeling the varying length of sampling intervals:

$$d^2_{STS}(X, X') = \sum_{i=1}^{t_X - 1} \left(\frac{X_{i+1} - X_i}{t_{i+1} - t_i} - \frac{X'_{i+1} - X'_i}{t_{i+1} - t_i} \right)^2.$$

For the Euclidean distance and the STS distance, the series need to have the same phase and length, which is often not the case in practice.

Probability-based measures view time series as probability distributions. Probability-based distance (Kumar et al., 2002) is based on the distribution of errors data, which they model through a Gaussian distribution. The resulting distance follows a Chi-square distribution and is scale invariant. Kullback-Leibler (KL) divergence (Warren Liao, 2005) for discrete distributions can be readily applied to time series. However, important differences in phase between the time series result in significant value differences between probability distributions. Moreover, KL divergence imposes that the two time series have the same length.

2-dimensional singular value decomposition (2dSVD) (Weng & Shen, 2008) uses the eigenvectors of row-row and column-column covariances as features, scoring the distance between two time series with the distance between these features. Locality preserving projections (LPP) (Weng & Shen, 2008) extends 2dSVD by trying to find a low-dimensional projection space for the features extracted with 2dSVD where samples from the same class are close. Both 2dSVD and LPP are sensitive to noisy data and outliers.

Complexity-invariant distance (Batista et al., 2014) uses information about complexity differences between two time series as a correction factor for existing distance measures. For example, the Euclidean distance can be made complexity-invariant through the following correction:

$$d_{CID}(X, X') = d_2(X, X') \cdot \frac{\max(CE(X), CE(X'))}{\min(CE(X), CE(X'))},$$

where CE is a complexity estimate defined as $CE(X) = \sqrt{\sum_{t=1}^{t_X - 1} (X_t - X_{t+1})^2}$. Compression rate distance (Vinh & Anh, 2015) is based on Batista et al. (2014), using the notion of compression rate instead of the complexity factor. The compression rate is computed based on the notion of entropy. The higher its value between two time series is, the closer

they should be. Both distances can be computed efficiently in linear time in the length of the time series.

Measuring the distance or similarity between time series that have the same length and are sampled uniformly is not a complex problem. However, in time series applications, the data is usually sampled by different sensors at different frequencies. In this case, there is no one-to-one correspondence between the time moments, and temporal distortions and shifts should be taken into account. To this end, time series need to be realigned before being compared.

Time series alignment Given two univariate or multivariate time series X and X', an alignment π is expressed as a sequence of pairs of indices:

$$\pi = \begin{pmatrix} \pi_X(k) \\ \pi_{X'}(k) \end{pmatrix}, \quad k = 1, \dots, t_{XX'},$$

where $\pi_X(k)$ is an index from time series X, $\pi_{X'}(k)$ is an index from X', and $t_{XX'}$ is the length of the alignment π. The pair of indices indicates that time moment X_i of index $\pi_X(k)$ corresponds to time moment X'_j of index $\pi_{X'}(k)$. Under an alignment, the two time series X and X' can be extended to two new ones $\bar{X}$ and $\bar{X}'$, expressed as

$$\begin{cases} \bar{X}_k = X_i \\ \bar{X}'_k = X_j \end{cases}, \quad k = 1, \dots, t_{XX'}.$$

$\bar{X}$ and $\bar{X}'$ now have the same length and can be compared directly (e.g. through Euclidean distance). The warping path is correctly constructed if it satisfies the following constraints:

- Boundary constraint: $\pi_1 = (1, 1), \pi_{t_{XX'}} = (t_X, t_{X'})$;
- Monotonicity: if $\pi_k = (i, j)$ and $\pi_{k+1} = (i', j')$, then $i' \geq i$ and $j' \geq j$, that is the alignment cannot progress backwards in time;
- Continuity: if $\pi_k = (i, j)$ and $\pi_{k+1} = (i', j')$, then $i' \leq i + 1$ and $j' \leq j + 1$, that is every point in the time series must be used in the alignment.

Under the previous constraints, an alignment is always at least as long as the longest of the two time series, but shorter than their cumulated lengths, i.e. $t_{XX'} \in [\max(t_X, t_{X'}), t_X + t_{X'}]$. In this dissertation, we will always consider alignments constructed with these properties. We now present the most well-known algorithm for computing the optimal alignment for a given pair of time series.

Dynamic time warping Dynamic time warping (DTW) (Kruskall & Liberman, 1983) is an elastic measure of the similarity between two time series which seeks to provide the best alignment between them. Its popularity is due to its capacity to work with series of different lengths and phases. DTW uses a dynamic programming technique to find the minimal distance between two time series, where sequences are warped by stretching or

Algorithm 3.1 Dynamic time warping

Input Time series X of length t_X, time series Y of length t_Y, distance $d(\cdot, \cdot)$
Output Distance between X and Y

```
DTW := array(m, n)
for i = 1 to t_X do
    DTW[i, 0] ← ∞
end for

for j = 1 to t_Y do
    DTW[0, j] ← ∞
end for

DTW[0, 0] ← 0
for i = 1 to t_X do
    for j = 1 to t_Y do
        DTW[i, j] ← d(X[i], Y[j]) + min(DTW[i − 1, j], DTW[i, j − 1], DTW[i − 1, j − 1])
    end for
end for
return DTW[m, n]
```

shrinking the time dimension (see Figure 3.3). The method produces a valid alignment and can be summarized as follows. Given a pair of time series X and X', the first step is to construct a cost matrix $\mathbf{C} \in \mathbb{R}^{t_X \times t_{X'}}$ where the entry at indices (i, j) represents the cost of aligning the time moment i from series X to the time moment j in X'. The cost matrix can be known or fixed for certain tasks, but in most of the cases it is computed using a distance (usually, the Euclidean distance). Using $\mathbf{C}$, DTW will compute a cumulative cost matrix $\mathbf{C}^+ \in \mathbb{R}^{t_X \times t_{X'}}$, where each element $\mathbf{C}^+_{i,j}$ represents the minimum cost for aligning series X up to point i with series X' up to point j. This minimum cumulative cost is obtained using the relationship:

$$\mathbf{C}^+_{i,j} = \mathbf{C}_{i,j} + \min(\mathbf{C}^+_{i-1,j-1}, \mathbf{C}^+_{i,j-1}, \mathbf{C}^+_{i-1,j}), \tag{3.2}$$

where $\mathbf{C}^+_{1,1} = \mathbf{C}_{1,1}$. After computing all the elements in $\mathbf{C}^+$, the minimum distance under the best alignment is found in $\mathbf{C}^+_{t_X,t_{X'}}$, and the alignment can be recovered through backtracking. The pseudo-code of DTW is detailed in Algorithm 3.1.

DTW depends on the local distance defining the cost matrix used for the alignment. Using a task-specific cost matrix $\mathbf{C}$ can greatly improve performance of DTW. Notice that using a similarity (or affinity) matrix instead of a cost matrix transforms the computation of DTW in Equation (3.2) in a maximization problem. DTW is an elastic measure, it thus does not have distance properties, even when the cost matrix $\mathbf{C}$ is constructed using a proper distance.

The extensive comparative study from (Ding et al., 2008) proved that DTW is among the best measures for comparing time series and that the accuracy of the Euclidean distance converges to DTW as the size of the training set increases. Its major drawback is the quadratic complexity in the length of the time series. These two reasons have motivated an important research effort for improving the results of naive DTW, either by proposing

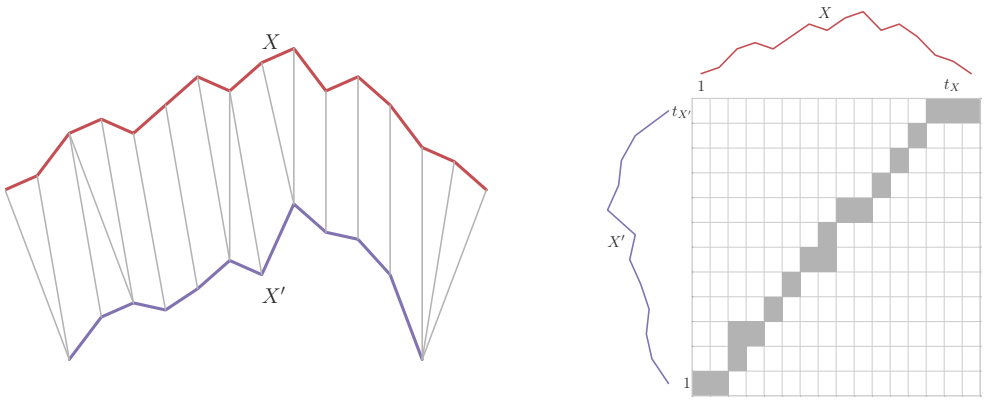
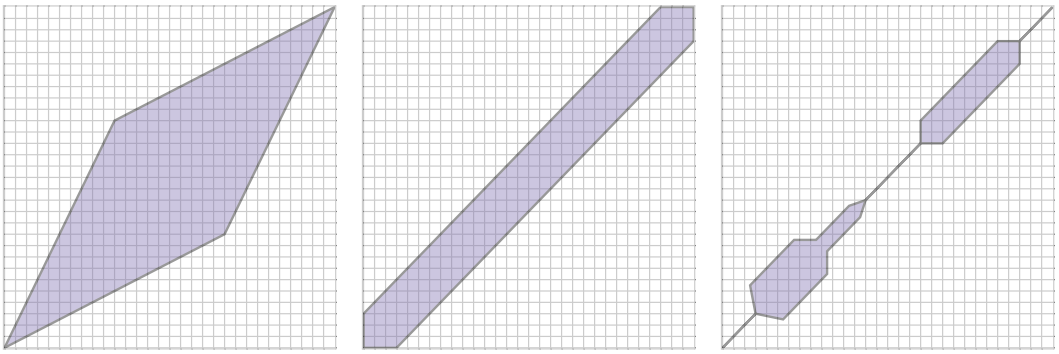


Figure 3.3: Example of dynamic time warping alignment.

efficient computation methods (Al-Naymat et al., 2009; Salvador & Chan, 2004) that reduce complexity, or by adjusting the constraints on the time deformation that is allowed.

Alignment constraints for DTW Applying additional constraints in the construction of the alignment has two main motivations: (i) it can speed up the computation by analyzing only a subset of all the possible warping paths and (ii) it can help avoid pathological warping (e.g. aligning the beginning of a time series with the end of another). Most heuristics for global constraints define a maximum range, usually around the diagonal, where the alignments are allowed. The Itakura parallelogram (Itakura, 1975) present a window which starts with width one at both ends of the time series and increases linearly from both ends to the middle of the time series (see Figure 3.4(a)); warping is possible only inside this area. Sakoe-Chiba band (Sakoe & Chiba, 1978) was originally used for speech recognition and is now one of the most popular global path constraints. It imposes a band of constant width around the diagonal in which warping is allowed (Figure 3.4(b)), and is usually specified as a percentage of the length of the time series. In practice, small widths ($<$10%) work best in most of the cases (Keogh & Ratanamahatana, 2004). Setting the width of the band to zero yields the Euclidean distance. Ratanamahatana & Keogh (2004) proposed the Ratanamahatana-Keogh (R-K) band, an arbitrary shaped constraint computed from the data. The constraint can be written as a vector of values defining the allowed range of warping at each point of a time series, as depicted in Figure 3.4(c). The authors propose a heuristic to automatically find the most appropriate width at each time point and show that the band improves classification performance, but determining the form of the band is computationally more expensive than fixed-size windows. The Sakoe-Chiba band and the Itakura parallelogram are special cases of the R-K band. Yu et al. (2011) propose a large margin criterion to find the optimal bands which best separate the classes. Similarly to the R-K band, here also the constraint areas have variable widths depending on the time axis. The solution is found through brute force calculations, making the approach rather inefficient.

The common assumption on which most of the previous methods are based is that time series from the same class share a global pattern, while different classes are distinguishable on at least local some shapes. To handle the cases where this condition is not satisfied,



(a) Itakura parallelogram. (b) Sakoe-Chiba band. (c) R-K band.

Figure 3.4: Global constraints for DTW.

Frambourg et al. (2013) propose a method driven by within-class variance minimization and between-class variance maximization for k-NN classification of multivariate time series. They generalize the notion of variance for multivariate time series. The purpose it to learn a matching matrix $\mathbf{M}$ per time series, so as to connect time series based on their discriminative features. This is done through an iterative approach: the algorithm considers the set of all possible matches (links) between time moments, for which a notion of utility w.r.t. the variance is computed; the links with too small a contribution are deleted. The utility computation and links deletion steps are iterated until all links that fit the criterion are removed. This procedure is performed for minimizing intra class variance and maximizing inter class variance. The weighting matrices $\mathbf{M}$ are used when comparing new examples to the training set in k-NN classification.

Empirical Mahalanobis distance The method introduced in (Prekopcsák & Lemire, 2012) proposes Mahalanobis-based distances for univariate time series k-NN classification. They consider the original form of the Mahalanobis distance, based on the inverse covariance matrix of the data. As this matrix is not always invertible (i.e. when the number of samples is lower than the length of the time series), they use three heuristics for approximating it: a pseudoinverse, a covariance shrinkage method and a diagonal form of the matrix. The experiments show that these heuristics perform worse than the classic DTW, but have the advantage of being faster to compute.

Sequence alignment The problem of aligning time series is strongly related to the more general one of sequence alignment. The latter is of impact in many applications in bioinformatics, like DNA and protein sequencing, as well as handwriting recognition, spell-checkers and natural language translation. Some of the measures used to compare sequences, more precisely character strings, have been adapted to time series comparison. String *edit distance* (Levenshtein, 1966) is a distance between strings of possibly different lengths built from a given alphabet. It is based on three types of operations: insertion, deletion and substitution of a symbol, each of which has a specific cost. A sequence of operations transforming a string into another is called an *edit script*. The edit distance between two strings is defined as the cost of the cheapest edit script that turns one string

into the other. Computing the edit distance is done through a dynamic programming algorithm similar to DTW, that also has quadratic complexity. The Levenshtein distance is a particular form of edit distance which uses a unit cost matrix, thus corresponding to the minimum number of operations turning one string into another. The edit distance has been adapted to real sequences (Chen et al., 2005), making it readily applicable to time series. Longest common subsequence (LCS) metric (Hirschberg, 1975; Maier, 1978) is also related to the edit distance, but it allows only insertion and deletion, not substitution. Its application to time series (Vlachos et al., 2004) is based on the idea that the longer common subsequences the two time series have, the closer they are. The most desirable characteristic in this distance is that it can waive noise and distortions in time series.

Temporal kernels Bahlmann et al. (2002) proposed a kernel for time series based on the Euclidean distance and an optimal alignment between X and X':

$$K_{eucl}(X, X') = \exp\left(- \min_{\pi \in \mathcal{A}(X,X')} \frac{1}{t_{XX'}} \sum_{i=1}^{t_{XX'}} \left\| X_i - X'_i \right\|^2 \right).$$

Another temporal kernel was proposed by Shimodaira et al. (2002), based directly on the Gaussian kernel, also under an optimal alignment of the time series:

$$K_{gaus}(X, X') = \max_{\pi \in \mathcal{A}(X,X')} \frac{1}{t_{XX'}} \sum_{i=1}^{t_{XX'}} \exp(-\frac{1}{\sigma^2} \left\| X_i - X'_i \right\|^2).$$

The exponentiation in the two previous formulations is introduced to make the kernels positive definite, but this property is not guaranteed neither in theory, nor in practice. Global alignment (GA) kernel Cuturi (2011); Cuturi et al. (2007) is in turn not based on an optimal alignment under a certain criterion. It is instead defined as the exponentiated soft-minimum of all possible alignment distances,

$$K_{\mathrm{GA}}(X, X') = \sum_{\pi \in \mathcal{A}(t_X, t_{X'})} \exp(-d_{X,X'}(\pi)).$$

In the sense of the kernel K_{GA}, two sequences are similar if they share a wide set of efficient alignments. Cuturi et al. (2007) show that the GA kernel is positive definite under mild conditions, while Cuturi (2011) proposes an efficient method for computing it.

Most of the metrics presented in this section, including DTW, are not implicitly designed to handle time series with multiple features. One way of adapting these methods is to weigh features equally, but that does not take into account the importance of each feature, nor the possible differences in scale. ten Holt et al. (2007) proposed an adaptation of DTW to the multivariate case. Shokoohi-Yekta et al. (2015) argue that DTW can be computed in two different ways: (i) when the features are strongly correlated, first compute a score for each time moment based on all features, then compute the DTW score for the whole time series using the local scores, or, (ii) when the features are only loosely coupled, compute the DTW score of each feature independently (treating them as univariate time series), then combine them in a global DTW score. They show that in practice there is not one of

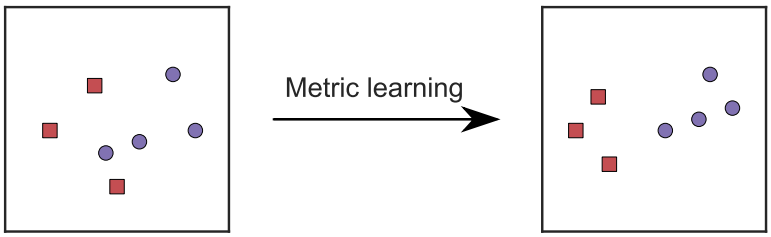


Figure 3.5: Intuition behind metric learning.

these methods that outperforms the other, but that the choice must be made depending on the application.

3.3 Metric Learning

Nonparametric distances or similarities do not have the capacity to adapt to the specificities of the problem, thus might not yield the best results. Moreover, the choice of metric has a crucial impact on performance.

In the previous sections, we have seen some parameterizable metrics which can be adapted to the problem at hand. Manually tuning these metrics can be a difficult task, especially when the number of parameters is elevated, or there is not much prior information available about the task. Metric learning aims at finding the parameters of a distance or similarity function that best account for the underlying geometry of the data. This section focuses on giving a comprehensive image of the state of the art in supervised and semi-supervised metric learning. For more details about these fields and some other settings that are not covered in this document, see the surveys from Bellet et al. (2013, 2015); Kulis (2013); Yang (2006).

In most instances of (semi-)supervised metric learning, the criterion that the learned metric should satisfy enforces class structure: examples from the same class are considered similar (or close in terms of distance) and should be placed together, while examples from different classes are dissimilar (far) and should not be mixed by the metric (see Figure 3.5). The parameters of the distance or similarity function are learned under the constraints induced by the previous intuition. The constraints from the data are usually integrated as weak supervision over pairs or triplets of points, with different semantics:

- Positive pairs (or must-link constraints): $\mathcal{P} = \{(z, z') \in \mathcal{S} \times \mathcal{S}$, where z and z' should be similar$\}$;
- Negative pairs (or cannot-link constraints): $\mathcal{N} = \{(z, z') \in \mathcal{S} \times \mathcal{S}$, where z and z' should be dissimilar$\}$;
- Relative triplets: $\mathcal{R} = \{(z, z', z'') \in \mathcal{S} \times \mathcal{S} \times \mathcal{S}$, where z is more similar to z' than to $z''\}$. Triplet constraints are a weaker form of supervision than pairs, thus being in many applications more easily available.

In a supervised setting, these constraints are easy to generate directly from the class information of the data. Nevertheless, producing all the pairs or triplets is costly and can yield an extremely large amount of constraints to satisfy, as the number of pairs (triplets) is quadratic (cubic) in the number of examples. On the other hand, the metric can be learned based only on a subset of constraints, but then the question of which pairs or triplets are more representative and how to find them becomes essential.

Under the constraints, the problem of learning the parameters of a metric function (usually the entries of a matrix) can be formulated as an optimization problem over a loss function in a regularized setting:

$$\hat{\mathbf{M}} = \underset{\mathbf{M}\in\mathbb{R}^{d\times d}}{\arg\min}\, \ell(\mathbf{M}, \mathcal{P}, \mathcal{N}, \mathcal{R}) + \lambda\, \mathrm{reg}(\mathbf{M}).$$

The methods from the state of the art are differentiated by the choice of the loss function ℓ, the regularizer $\mathrm{reg}(\cdot)$, and the constraints they use $(\mathcal{P}, \mathcal{N}, \mathcal{R})$, as we will see in the following.

3.3.1 Metric Learning for Feature Vectors

In the case of feature vectors, the instances are represented as vectors from a space $\mathcal{X} \subseteq \mathbb{R}^d$. The methods in this category usually do not natively support categorical features. In this section, we review methods for supervised Mahalanobis distance learning (Section 3.3.1.1), supervised similarity learning (Section 3.3.1.2), semi-supervised metric learning (Section 3.3.1.3) and metric learning with generalization guarantees (Section 3.3.1.4). An overview of the other types of settings for learning metrics is given in Section 3.3.1.5.

3.3.1.1 Mahalanobis Distance Learning

Xing et al. Xing et al. (2002) proposed the first Mahalanobis distance learning method. Designed for clustering, the formulation aims at maximizing the distance between dissimilar points, while keeping the one between similar points under a certain threshold.

$$\begin{aligned} \max_{\mathbf{M}\in\mathbb{S}_+^d} & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{N}} d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{P}} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \leq 1, \end{aligned} \tag{3.3}$$

where $\mathbb{S}_+^d$ the cone of symmetric PSD $d\times d$ real-values matrices. Problem (3.3) is convex and solved by a semi-definite programming (SDP) approach based on eigenvalue decomposition (cubic complexity in the number of features). This makes it intractable for medium and high-dimensional problems. Meanwhile, the lack of regularization makes it also sensitive to overfitting.

Schultz & Joachims Schultz & Joachims (2003) learn a Mahalanobis distance $\mathbf{M}$ from a set of relative constraints, under the additional assumption that $\mathbf{M} = \mathbf{A}^T\mathbf{W}\mathbf{A}$. Here, $\mathbf{A}$ is a fixed matrix, and $\mathbf{W}$ is diagonal. The squared Mahalanobis distance becomes

$$d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{A}\mathbf{x}_i - \mathbf{A}\mathbf{x}_j)^T\mathbf{W}(\mathbf{A}\mathbf{x}_i - \mathbf{A}\mathbf{x}_j).$$

The form of $\mathbf{M}$ ensures that it is PSD. $\mathbf{W}$ can be learned by solving the following quadratic program:

$$\begin{aligned} &\min_{\mathbf{W}} \|\mathbf{M}\|^2_{\mathcal{F}} \\ &\text{s.t.} \quad d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1, \forall (z_i, z_j, z_k) \in \mathcal{R}, \end{aligned} \tag{3.4}$$

where $\|\cdot\|^2_{\mathcal{F}}$ is the squared Frobenius norm. This formulation is then adapted to allow margin violations by introducing slack variables. As the method only learns the diagonal matrix $\mathbf{W}$, the metric is limited to a weighting of the Euclidean distance. Moreover, handpicking $\mathbf{A}$ can be difficult; in practice, it is set to $\mathbf{I}$.

Goldberger et al. Neighborhood Component Analysis (NCA) (Goldberger et al., 2004) learns a Mahalanobis distance that minimizes the expected leave-one-out error of the k-NN classifier. For this, they use the Cholesky decomposition of the matrix $\mathbf{M} = \mathbf{L}^T\mathbf{L}$. They define the probability that two points $\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors using a softmax rule over the Euclidean distance in the transformed space:

$$p_{ij} = \frac{\exp(-\|\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j\|^2)}{\sum_{k \neq i} \exp(-\|\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_k\|^2}, \quad p_{ii} = 0.$$

The distance is learned over $\mathbf{L}$ such that it maximizes the probability to correctly classify $\mathbf{x}_i$ for all i:

$$\max_{\mathbf{L}} \sum_i \sum_{j : y_i = y_j} p_{ij}. \tag{3.5}$$

When $\mathbf{L}$ is nonsquare, choosing $\mathbf{L} \in \mathbb{R}^{d \times d'}$ with $d' < d$, induces a low-rank metric and allows to project the data in a lower dimensional space. The main limitation of solving Equation (3.5) over $\mathbf{L}$ is that the problem is nonconvex, thus being subject to local maxima. Moreover, d' has to be selected by hand.

Shalev-Shwartz et al. POLA (Shalev-Shwartz et al., 2004) is the first online method for learning a Mahalanobis distance from positive and negative pairs. The method learns the parameter matrix $\mathbf{M}$, as well as a threshold $b \geq 1$, used to separate classes, expressed for a tuple (z_i, z_j) as:

$$y_i y_j (b - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)) \geq 1.$$

The formulation considers one pair at a time and updates the parameters in two projection steps. The first step projects onto the set of admissible solutions for the current examples under hinge loss for large margin separation. This is obtained using an efficient closed-form solution. The second step projects the current solution matrix back onto the PSD cone. Here, the projection is done efficiently, as it only needs to compute the minimum eigenvalue

instead of a full eigenvalue decomposition (cubic complexity in the number of dimensions of the data). The online algorithm comes with a regret bound for the separable case, and POLA also has batch version.

Globerson & Roweis Maximally collapsing metric learning (MCML) (Globerson & Roweis, 2006) learns a Mahalanobis distance using the same probability distribution p_{ij} as in NCA, but this time optimizing directly over $\mathbf{M}$. The idea is to consider the best representation of the data to be of all members of a class as a single point, making the distance between them zero, while placing different classes at infinite distance. Their objective function minimizes the Kullback-Leibler divergence between p_{ij} and the ideal mapping. The objective being convex in $\mathbf{M}$, MCML does not have the difficulty of local optima as NCA. Instead, it suffers from the same expensive projections as MMC.

Weinberger & Saul Large margin nearest neighbor (LMNN) (Weinberger & Saul, 2008, 2009; Weinberger et al., 2006) is one of the most popular and effective methods for learning Mahalanobis distances for the k-NN classifier. The intuition is to obtain neighborhoods where all k points belong to the same class, while pushing instances from other classes outside the neighborhood. The method is based on positive and relative constraints, redefined through the notion of neighborhood:

$$\mathcal{P} = \{(z_i, z_j) \in \mathcal{S} \times \mathcal{S}, \text{ where } y_i = y_j \text{ and } \mathbf{x}_j \text{ is in the } k\text{-neighborhood of } \mathbf{x}_i\},$$
$$\mathcal{R} = \{(z_i, z_j, z_k) \in \mathcal{S} \times \mathcal{S} \times \mathcal{S}, \text{ where } (z_i, z_j) \in \mathcal{P} \text{ and } y_i \neq y_k\}.$$

The distance matrix $\mathbf{M}$ is learned by solving a convex problem:

$$\min_{\mathbf{M} \in \mathbb{S}_+^d, \xi_{ijk} \geq 0} (1-\mu) \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + \mu \sum_{i,j,k} \xi_{ijk}$$
$$\text{s.t.} \quad d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \xi_{ijk} \quad \forall (i,j,k) \in \mathcal{R},$$

where $\mu \in [0,1]$ is a trade-off parameter, and ξ_{ijk} are slack variables that allow soft constraints. A specific solver based on subgradient descent and sets of active constraints allows the method to scale to millions of triplets. In spite of its lack of regularization, LMNN performs well in most cases and has been the basis of many variants. Bellet et al. (2012) have shown that LMNN is prone to overfitting for high-dimensional data.

Davis et al. Information-theoretic metric learning (ITML) (Davis et al., 2007) introduces LogDet divergence regularization for Mahalanobis distance learning, at the same time providing a cheap way for ensuring that $\mathbf{M}$ stays PSD. This Bregman divergence for PSD matrices is defined as:

$$D_{ld}(\mathbf{M}, \mathbf{M}_0) = \operatorname{tr}(\mathbf{M}_0^{-1}) - \log\det(\mathbf{M}_0^{-1}) - d,$$

where $\mathbf{M}_0$ is a preset PSD matrix to which $\mathbf{M}$ should remain close, acting as a regularizer. In practice, $\mathbf{M}_0$ is usually set to the identity matrix, implying that $\mathbf{M}$ stays close to

the Euclidean distance. The LogDet divergence is finite if and only if $\mathbf{M}$ is PSD, thus minimizing it yields a proper pseudo-distance. ITML is formulated as follows:

$$\begin{aligned} &\min_{\mathbf{M}\in\mathbb{S}_+^d} D_{ld}(\mathbf{M},\mathbf{M}_0) \\ &\text{s.t.} \quad d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \leq u \quad \forall(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S} \\ &\qquad d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_k) \geq v \quad \forall(\mathbf{x}_i,\mathbf{x}_k)\in\mathcal{D}, \end{aligned} \tag{3.6}$$

where $u, v \in \mathbb{R}$ are parameters for distance limits. Slack variables are further introduced to allow violations. This convex problem is solved efficiently, avoiding eigenvalue decomposition and semi-definite programming. Intuitively, minimizing $D_{ld}(\mathbf{M},\mathbf{M}_0)$ is equivalent to reducing the KL divergence between two Gaussian distributions characterized by $\mathbf{M}$ and $\mathbf{M}_0$. Finally, $\mathbf{M}$ strongly depends on the initial value $\mathbf{M}_0$, which is an important shortcoming, as $\mathbf{M}_0$ is handpicked.

Wang et al. More recently, Wang et al. (2012b) designed a generic Mahalanobis distance learning framework for k-NN classification, which can be cast into well-known methods like MMC, MCML and LMNN. For this, they use a Laplacian matrix $\mathbf{P} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{S}|}$ modeling the target neighbor relationships: $P_{ij} = 1$, if $\mathbf{x}_j$ is a neighbor of $\mathbf{x}_i$, otherwise, $P_{ij} = 0$. In addition to learning the matrix $\mathbf{M}$ parameterizing the Mahalanobis distance, this method also learns the target neighborhood to be considered, i.e. the constraints that work best for the problem. Their problem can be written as:

$$\begin{aligned} &\min_{\mathbf{M},\mathbf{P},\Xi} \sum_{i,j} P_{ij}\cdot f_{ij}(\mathbf{M},\Xi) \\ &\text{s.t.} \sum_{i,j} = |\mathcal{S}|\cdot K_{avg} \\ &\qquad K_{max} \geq \sum_j P_{ij} \geq K_{min} \\ &\qquad 1 \geq P_{ij} \geq 0, \\ &\qquad \text{constraints from the original problem,} \end{aligned}$$

where Ξ are the parameters specific to the original problem and f_{ij} is the function relating the parameters $\mathbf{M}$ and Ξ to the neighborhood P_{ij} in the original setting. Here, $K_{max} \geq K_{avg} \geq K_{min}$ are respectively the maximum, average and minimum number of neighbors an instance can have, and are set in advance. The problem is solved by alternating learning steps over the metric and the neighborhood, both of which are convex problems, provided that the initial metric learning approach is convex.

Huo et al. Huo et al. (2016) introduce capped trace norm regularization for metric learning. Its purpose is to provide a low-rank metric without using Cholesky decomposition or having to manually tune the rank of the matrix $\mathbf{M}$. The capped trace norm is defined as:

$$\text{reg}(\mathbf{M}) = \sum_{i=1}^{d} \min\{\sigma_i, \epsilon\},$$

where σ_i represent the singular values of the learned matrix, and ϵ is a fixed threshold. The impact of this regularizer is that it only minimizes the singular values which are lower than ϵ. The method uses quadruplet constraints (Law et al., 2013), which encompass the standard pair and triplet constraints and are expressed as:

$$\mathcal{A} = \{(z_i, z_j, z_k, z_l) \in \mathcal{S} \times \mathcal{S} \times \mathcal{S} \times \mathcal{S} : d^2_{\mathbf{M}}(x_k, x_l) \geq d^2_{\mathbf{M}}(x_i, x_j)\}.$$

The constraints are used to solve the following problem:

$$\min_{\mathbf{M} \in \mathbb{S}^d_+} \sum_{q \in \mathcal{A}} \left[\xi_q + (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j) - (\mathbf{x}_k - \mathbf{x}_l)^T \mathbf{M} (\mathbf{x}_k - \mathbf{x}_l)\right]_+ + \frac{\lambda}{2} \operatorname{reg}(\mathbf{M}),$$

where λ is the regularization parameter and ξ_q are the margins for each quadruplet. The formulation is non-smooth and non-convex, so the authors solve a convex surrogate instead. However, using this approach, the final solution only converges to a local minima.

3.3.1.2 Similarity Learning

Qamar & Gaussier SiLA (Qamar & Gaussier, 2009b; Qamar et al., 2008) is an extension of the voted perceptron algorithm allowing to learn a generic similarity function of the form:

$$K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^T \mathbf{M} \mathbf{x}'}{N(\mathbf{x}, \mathbf{x}')},$$

where N is a normalization depending on $\mathbf{x}$ and $\mathbf{x}'$. Like in the case of most similarity functions, $\mathbf{M} \in \mathbb{R}^{d \times d}$ is a square matrix that is not necessarily PSD, nor symmetric. At each iteration, the algorithm proceeds to update the matrix in an online manner using the perceptron rule for the points that do not respect the separation criterion. This criterion is based on target neighborhoods similar to those from LMNN, but here the neighbors are recomputed between iterations under the new $K_{\mathbf{M}}$. The method comes with regret bounds for the separable and nonseparable settings, based on the theory of the voted perceptron.

Qamar & Gaussier gCosLA (Qamar & Gaussier, 2009a) is an online method for learning generalized cosine similarities defined as:

$$K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^T \mathbf{M} \mathbf{x}'}{\sqrt{\mathbf{x}^T \mathbf{M} \mathbf{x}} \sqrt{\mathbf{x}'^T \mathbf{M} \mathbf{x}'}},$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is symmetric and PSD. Positive pairs are used to guide the method under a hinge loss function. The algorithm is a two steps approach, similar to POLA: a projection to obtain zero loss on the current pair, followed by a second one onto the cone of PSD matrices. The authors derive a closed-form solution for the former step, but the latter needs a full eigenvalue decomposition, making it computationally expensive. In practice, gCosLA improves performance over SiLA and has comparable results to LMNN and ITML. gCosLA also comes with regret bounds and a batch version.

Chechik et al. OASIS (Chechik et al., 2009) is an online approach for learning a bilinear similarity for image retrieval based on k-NN. The method uses relative constraints and a hinge loss over the triplets. Starting with $\mathbf{M} = \mathbf{I}$, each iteration t is performed by solving the following convex problem on one triplet drawn randomly:

$$\begin{aligned} \mathbf{M}^t = \underset{\mathbf{M},\xi}{\arg\min} \frac{1}{2} \left\|\mathbf{M} - \mathbf{M}^{t-1}\right\|_{\mathcal{F}}^2 + c\xi \\ \text{s.t. } 1 - d_{\mathbf{M}}^2(\mathbf{x}_i, \mathbf{x}_j) + d_{\mathbf{M}}^2(\mathbf{x}_i, \mathbf{x}_k) \leq \xi \quad \xi \geq 0, \end{aligned} \tag{3.7}$$

where c is a trade-off parameter that controls how much the solution will be affected by each iteration, and ξ is a slack variable. If the triplet constraint is satisfied, then no update is performed; otherwise, the solution is updated through a closed-form update based on the passive-aggressive algorithm (Crammer et al., 2006). As a result of using a bilinear similarity, the method can handle sparse data (in their experiments, images stored through bag-of-words) in an efficient way, allowing it to scale to millions of examples.

3.3.1.3 Semi-Supervised Metric Learning

The following metric learning methods use a semi-supervised setting in order to improve the performance through the use of unlabeled data.

Hoi et al. Laplacian regularized metric learning (LRML) (Hoi et al., 2008, 2010) is one of the first methods to explore unlabeled data explicitly for distance learning. It learns a Mahalanobis distance $\mathbf{M}$ with manifold regularization using a Laplacian matrix which integrates neighborhood relations. The side information comes in the form of positive and negative pairs. LRML is formulated as:

$$\begin{aligned} &\min_{\mathbf{M} \in \mathbb{S}_+^d} \operatorname{tr}(\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{M}) + \gamma_s \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}} d_{\mathbf{M}}^2(\mathbf{x}_i, \mathbf{x}_j) - \gamma_d \sum_{(\mathbf{x}_i, \mathbf{x}_k) \in \mathcal{N}} d_{\mathbf{M}}^2(\mathbf{x}_i, \mathbf{x}_j) \\ &\text{s.t.} \quad \log\det(\mathbf{M}) \geq 0, \end{aligned} \tag{3.8}$$

where $\mathbf{X}$ is the data matrix where each $\mathbf{x}^T$ is a row, $\gamma_d, \gamma_s > 0$ are trade-off parameters and tr is the trace operator. Here, $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is a Laplacian regularizer, where $\mathbf{W} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{S}|}$ is the Laplacian matrix and $\mathbf{D} \in \mathbb{N}^{|\mathcal{S}|\times|\mathcal{S}|}$ is a diagonal matrix computed as $D_{ii} = \sum_j W_{ij}$, i.e. containing the total number of neighbors for each point. The constraint is introduced to prevent trivial solutions where the entire space is shrunken ($\mathbf{M} = 0$). Problem (3.8) is solved either as a classic SDP or in a modified, faster version through matrix inversion. LRML is used to solve image retrieval and image clustering applications, with particularly good results compared to fully supervised methods when side information is scarce. This indicates that semi-supervised metric learning can leverage additional information coming from unlabeled data for better performance.

Zha et al. Log-determinant regularized distance metric learning L-DML (Zha et al., 2009) uses a similar formulation to that of LRML, with the distinction that the regularization term is a weighted sum using multiple metrics, which are learned over various datasets different

from the target task. The method learns one Mahalanobis distance using the positive and negative pairs from the given dataset to measure the loss, while the regularization term uses unlabeled data from different tasks.

$$\begin{aligned}\min_{\mathbf{M}\in\mathbb{S}_+^d,\boldsymbol{\alpha}} & \sum_{k=1}^{K}\alpha_k \operatorname{tr}(\mathbf{X}\mathbf{L}_k\mathbf{X}^T\mathbf{M}) + \gamma_s \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{P}} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) - \gamma_d \sum_{(\mathbf{x}_i,\mathbf{x}_k)\in\mathcal{N}} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) + \beta\,\|\mathbf{M}\|_2^2 \\ \text{s.t.} \quad & \sum_{k=1}^{K}\alpha_k = 1, \alpha_k \geq 0, \forall k,\end{aligned} \tag{3.9}$$

where $\boldsymbol{\alpha}$ contains the weights associated with each of the K metrics, β is a regularization parameter, and all the other notations correspond to the ones in LRML (Problem (3.8)). The authors propose to solve the problem by alternating optimization steps over $\mathbf{M}$ (SDP) and $\boldsymbol{\alpha}$ (LP).

Niu et al. SERAPH (Niu et al., 2012) is a semi-supervised information-theoretic approach that learns a Mahalanobis distance. The metric is optimized to maximize the entropy over labeled similar and dissimilar pairs (in our notation, $\mathcal{P}$ and $\mathcal{N}$), and to minimize it over unlabeled data ($\mathcal{U}$). Let $p^{\mathbf{M}}_{ij}(y) = p^{\mathbf{M}}(y|\mathbf{x}_i,\mathbf{x}_j)$ be the probability of labeling $(\mathbf{x},\mathbf{x}')$ with label y, parameterized by the distance matrix $\mathbf{M}$. SERAPH is expressed as:

$$\max_{\mathbf{M},\kappa} \sum_{\mathcal{P}\cup\mathcal{N}} \ln p^{\mathbf{M}}_{ij}(y_{ij}) - \frac{\gamma}{2}\kappa^2 + \mu \sum_{\mathcal{U}}\sum_{y} p^{\mathbf{M}}_{ij} \ln p^{\mathbf{M}}_{ij}(y) - \lambda \operatorname{tr}(\mathbf{M}), \tag{3.10}$$

where μ,λ are trade-off parameters between supervision, unsupervised information and regularization, y_{ij} is the (shared) label of $\mathbf{x}_i$ and $\mathbf{x}_j$ and κ is a dual variable. The penalty presumes a Gaussian prior on the expected data moments, following the generalized maximum entropy principle (Jaynes, 1957). The formulation in Problem (3.10) is non-convex, being solved through an expectation-maximization (EM) algorithm. In experiments for k-NN classification, SERAPH slightly outperforms fully supervised methods like LMNN.

Semi-supervised metric and kernel learning has also been used to aid unsupervised learning, more precisely clustering, through small amounts of labeled data (Baghshah & Shouraki, 2009; Bilenko et al., 2004; Yeung & Chang, 2007; Yin et al., 2010).

In Chapter 4, our contribution will show a different way of exploiting unlabeled data in a semi-supervised classification setup.

3.3.1.4 Metric Learning with Generalization Guarantees

Although a large number of methods have been developed for (semi-)supervised metric learning, few studies analyze the theoretical properties of this type of algorithm. In the context of metric learning, and in contrast to the general machine learning setting, generalization guarantees can be addressed from two points of view:

- The *consistency* of the learned metric, i.e. the capacity of the metric to unseen examples and constraints; in practice, this amounts to bounding the difference between its performance on the training sample and on unseen examples.

- The *performance* of the associated classifier for the given task in terms of generalization error when using the learned metric.

Another important distinction between generalization bounds for machine learning in general and metric learning comes from the sampling of the data. All the theoretical frameworks presented in Section 2.7 make the assumption that the examples are independent and identically distributed. While this claim is often satisfied when using labeled examples directly, as we have seen, metric learning algorithms usually incorporate constraints on pairs and triplets of points. Even though the examples themselves are drawn i.i.d., this is not the case for the constraints. For this reason, standard theoretical frameworks cannot be applied directly to metric learning. However, the exist strategies that allow to make the pairs or triplets independent.

We now discuss the two questions of metric consistency and classifier performance based on the studies that have explored them.

Metric consistency Jin et al. (2009) study a generic distance learning formulation under Frobenius regularization and pair constraints:

$$\min_{\mathbf{M}\in\mathbb{S}_+^d} \frac{1}{|\mathcal{S}|^2} \sum_{(z_i,z_j)\in\mathcal{S}\times\mathcal{S}} \ell(d^2_{\mathbf{M}}, z_i, z_j) + \lambda \left\|\mathbf{M}\right\|_{\mathcal{F}}, \tag{3.11}$$

where $\lambda > 0$ is the regularization parameter. The loss function they use is:

$$\ell(d^2_{\mathbf{M}}, z_i, z_j) = f(y_i y_j (1 - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j))),$$

where f is convex and Lipschitz continuous. The paper adapts the framework of uniform stability (see Section 2.7.3) to the case of learning distances from pairs of examples. For their setting, they provide a generalization bound for the learned metric, but, as it is always the case for uniform stability, this limits their framework to Frobenius regularized formulations. The bound they deduce relates the true error of the learned metric R^ℓ_P to the empirical loss $R^\ell_{\mathcal{S}}$:

$$|R^\ell_P(\mathbf{M}) - R^\ell_{\mathcal{S}}(\mathbf{M})| \leq \frac{8L^2R^2}{\lambda n} + \left(\frac{8L^2R^4}{\lambda} + 4Ls(d) + 2g_0\right)\sqrt{\frac{\ln(2/\delta)}{2n}},$$

where the loss is L-lipschitz, $\sup_{\mathbf{x}} \left\|\mathbf{x}\right\|_2 \leq R$, $s(d) = \min\left(\sqrt{\frac{dg_0}{2\lambda}}, \eta(d)\right)$, g_0 is the maximum loss when $\mathbf{M} = \mathbf{0}$ and $\mathrm{tr}(\mathbf{M}) \leq \eta(d)$. In practice, Problem (3.11) is solved through an online approach, for which they provide regret bounds.

Bian & Tao (2011, 2012) develop an ERM framework for Mahalanobis distance learning under constraints. The generic loss function they use is close to the one from (Jin et al.,

2009):

$$\ell(d^2_{\mathbf{M}}, z_i, z_j) = f(y_i y_j (c - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j))),$$

with $c > 0$ a threshold parameter separating positive from negative pairs, and f a convex and Lipschitz continuous function. The framework is formulated as follows:

$$\min_{(\mathbf{M},c)\in\mathcal{Q}} \frac{1}{|\mathcal{S}|^2} \sum_{(z_i,z_j)\in\mathcal{S}\times\mathcal{S}} \ell(d^2_{\mathbf{M}}, z_i, z_j), \tag{3.12}$$

where the set of constraints $\mathcal{Q} = \{(\mathbf{M}, c) : 0 \preceq \mathbf{M} \preceq \mathbf{I}, 0 \leq c \leq a_0, a_0 > 0\}$ plays the role of limiting the complexity of the hypothesis. The theoretical results prove the consistency of the learned distance, i.e. its convergence to the optimal value, as well as a generalization bound. This framework is applicable to a large range of loss functions, of which the authors study the logarithmic loss and a smoothed version of the hinge loss. However, the ERM design of the problem prevents it from being used with regularization.

Cao et al. (2012) use the Rademacher complexity to derive consistency bounds for metrics learned under several matrix norm regularizers. The theoretical results cover the bilinear similarity and the Mahalanobis distance. Given a sample $\mathcal{S}$ of n instances, their learning formulation (using the Mahalanobis distance) is the following:

$$\min_{\mathbf{M}\in\mathbb{S}^d_+, b\in\mathbb{R}} \frac{1}{n(n-1)} \sum_{i,j\in\mathbb{N}_n, i\neq j} [1 + y_i y_j (d_{\mathbf{M}}(x_i, x_j) - b)]_+ + \lambda \|\mathbf{M}\|^2, \tag{3.13}$$

where b is an offset they learn at the same time as the metric, λ is the regularization parameter and $\|\cdot\|$ is a general matrix norm. The consistency bound derived for Problem (3.13) is the following:

$$R^\ell_P(\mathbf{M}, b) - R^\ell_{\mathcal{S}}(\mathbf{M}, b) \leq \frac{4\mathfrak{R}_n}{\sqrt{\lambda}} + \frac{4(3 + 2X_*/\sqrt{\lambda})}{\sqrt{n}} + 2(1 + X_*/\sqrt{\lambda})\sqrt{\frac{2\ln(1/\delta)}{n}},$$

where $\mathfrak{R}_n$ is the Rademacher complexity and $X_* = \sup_{\mathbf{x},\mathbf{x}'} \left\|(\mathbf{x} - \mathbf{x}')(\mathbf{x} - \mathbf{x}')^T\right\|_*$.

Bellet & Habrard (2015) adapt the notion of algorithmic robustness (presented in Section 2.7.4) to the metric learning setting, where data is accessed through pairs of points. Their results hold for any matrix norm regularizer. For a $(K, \epsilon(\cdot))$-robust algorithm, with probability $1 - \delta$, we have:

$$R^\ell_P(\mathbf{M}) - R^\ell_{\mathcal{S}}(\mathbf{M}) \leq \epsilon(\mathcal{S}) + 2B\sqrt{\frac{2K\ln 2 + 2\ln(1/\delta)}{n}},$$

where the loss function ℓ is bounded by B. Furthermore, they show that a weak notion of robustness is a necessary and sufficient condition for a metric learning algorithm to generalize.

Regressive virtual metric learning (RVML) (Perrot & Habrard, 2015) is a Mahalanobis distance learning method based on virtual points. These points are set beforehand and help setting the constraints for the metric: each training point is assigned a virtual point, to which the metric will try to bring it close. Assigning a single virtual point per example drastically reduces the number of constraints of the problem, but makes the strategy for

choosing virtual points vital for performance. The authors propose two heuristics for virtual points selection, one based on optimal transport (Villani, 2009), the other on choosing one virtual point per class and pushing all the points from the same class towards it. Once the virtual points are selected, RVML is formulated as a regularized least-squares problem over matrix $\mathbf{L} \in \mathbb{R}^{d \times d'}$ coming from the decomposition $\mathbf{M} = \mathbf{L}^T\mathbf{L}$:

$$\min_{\mathbf{L}} \frac{1}{|\mathcal{S}|} \|\mathbf{XL} - \mathbf{V}\|_{\mathcal{F}}^2 + \lambda \|\mathbf{L}\|_{\mathcal{F}}^2, \tag{3.14}$$

where $\mathbf{X}$ is the data matrix, $\mathbf{V}$ is the virtual points matrix, and λ is the regularization parameter. Problem (3.14) has a closed-form solution using matrix inversions, which limits its scaling capacities for high dimensional datasets. The matrix $\mathbf{M}$ is PSD by construction, thus avoiding expensive projections. RVML comes with a consistency bound based on uniform stability:

$$R_P^\ell(\mathbf{M}) - R_{\mathcal{S}}^\ell(\mathbf{M}) \leq \frac{8C_{\mathbf{v}}^2 C_{\mathbf{x}}^2}{\lambda n}\left(1 + \frac{C_{\mathbf{x}}}{\sqrt{\lambda}}\right)^2 + \left(\left(1 + \frac{16C_{\mathbf{x}}^2}{\lambda}\right) C_{\mathbf{v}}^2 \left(1 + \frac{C_{\mathbf{x}}}{\sqrt{\lambda}}\right)^2\right)\sqrt{\frac{\ln(1/\delta)}{2n}},$$

where the training examples are bounded w.r.t. a constant $\|\mathbf{x}\|_2 \leq C_{\mathbf{x}}$, and the landmarks are also bounded as $\|\mathbf{v}\|_2 \leq C_{\mathbf{v}}$. The authors further link their formulation to the general framework in Jin et al. (2009).

Classifier performance with learned metrics To the best of our knowledge, the problem of finding the link between the prediction performance and the properties of a learned metric has only been tackled for linear classification.

Similarity learning for linear classification (SLLC) (Bellet et al., 2012) uses the theory of (ϵ, γ, τ)-good similarity functions (Balcan & Blum, 2006; Balcan et al., 2008b) to ensure the performance of the classifier constructed using the learned metric. The framework of (ϵ, γ, τ)-good similarity functions is one of the first results that relates the properties of a similarity function to those of a linear classifier using it. The contributions in this thesis are strongly based on the (ϵ, γ, τ)-good framework; we will thus present it in more detail in Chapter 4. Bellet et al. (2012) propose to directly optimize the (ϵ, γ, τ)-goodness of a bilinear similarity under Frobenius regularization before plugging it in the linear classifier proposed by Balcan et al. (2008b). They also derive consistency bounds for the metric through uniform stability.

Guo & Ying (2013) extend the results from (Bellet et al., 2012) to several matrix norms using a Rademacher complexity analysis, based on techniques from (Cao et al., 2012). Recently, this framework has further been adapted to regularized online learning from pairs of points with theoretical guarantees (Guo et al., 2016).

3.3.1.5 Other Types of Approaches

Nonlinear metric learning The approaches we have analyzed so far learn a linear metric. Only a small number of methods have directly used nonlinear forms of metrics. The main difficulty is that introducing nonlinearity often makes problem suffer from non

convexity.

Chopra et al. (2005) proposed the first method for learning nonlinear metrics. They aim to learn a low-dimensional representation of the data where the examples in positive pairs are represented as close, while negative pairs are far. The nonlinear projection is obtained through a multilayer convolutional neural network. The weights of this model are learned through back-propagation and stochastic gradient descent, minimizing a loss function which imposes a separation criterion.

Gradient-boosted LMNN (GB-LMNN) (Kedem et al., 2012) is a version of LMNN learning a distance in a projection space determined by a nonlinear function ϕ. The mapping is defined as a combination between the metric of standard LMNN and additive combination of gradient boosted regression trees (Friedman, 2000). The algorithm is iterative and aims at adding to the solution a new tree from the set of all regression trees of fixed depth at each step.

Another way of introducing nonlinearity is to kernelize a linear method using the kernel trick, as it is often done for SVMs. Methods that we have presented earlier which have been kernelized include Bellet et al. (2012); Davis et al. (2007); Schultz & Joachims (2003); Shalev-Shwartz et al. (2004).

Multiple metric learning Learning multiple linear metrics has the capacity to capture the heterogeneities of complex tasks for which using one metric does not perform well. The purpose is often to learn one metric per region of the space, or even one metric per examples. The challenge in this setting becomes the computational complexity of learning the metrics simultaneously, as well as providing a coherent global metric, that varies smoothly and compares examples from different regions in a consistent way.

M^2-LMNN Weinberger & Saul (2008, 2009) is an extension of LMNN that learns multiple metrics. The training data is first partitioned in C clusters. M^2-LMNN learns one metric per cluster using the same objective as LMNN. When computing a distance, the metric associated to the first point in a pair is used. The overall learned metric is thus not symmetric, nor smooth. Moreover, it is sometimes prone to overfitting and computationally expensive.

Parametric local metric learning (PLML) (Wang et al., 2012a) learns one Mahalanobis distance per instance. Each distance is expressed as a linear combination of a limited number of metrics. The overall metric varies smoothly over the data manifold. The problem is solved in two steps: first, the weights for each instance are computed, then the metric bases are learned.

Sparse compositional metric learning (SCML) (Shi et al., 2014) learns a Mahalanobis distance in the form of a sparse linear combination of rank-one matrices. The combination of bases is custom per instance, but is still lightweight. The main disadvantage of the method is that the set of rank-one matrices is considered known (although the authors propose to generate them by clustering and Fischer discriminant analysis).

C2ML (Zantedeschi et al., 2016) addresses the problem of learning local combinations of

Table 3.1: Main characteristics of the reviewed metric learning methods for feature vectors.

Method	Convex	Scalable	Regularized	Semi-sup.	Online	Generalization
Xing et al.	✓	×	×	×	×	×
Schultz & Joachims	✓	✓	✓	×	×	×
POLA	✓	✓	×	×	✓	×
NCA	×	✓	×	×	×	×
MCML	✓	×	×	×	×	×
LMNN	✓	✓	×	×	×	×
Wang et al.	✓	✓	✓	×	×	×
ITML	✓	✓	✓	×	×	×
SiLA	-	✓	×	×	✓	×
gCosLA	✓	✓	×	×	✓	×
OASIS	✓	✓	✓	×	✓	×
LRML	✓	✓	✓	✓	×	×
L-DML	✓	✓	✓	✓	×	×
SERAPH	×	✓	✓	✓	×	×
Jin et al.	✓	✓	✓	×	✓	✓
Bian & Tao	✓	✓	×	×	×	✓
SLLC	✓	✓	✓	×	×	✓
RVML	✓	✓	✓	×	×	✓

metrics to obtain a smooth and symmetric overall metric. Starting from a partition of the space and a score function for each region, C2LM defines a metric between points as a weighted combination of the models. A weight vector is learned for each pair of regions, and a spatial regularization ensures that nearby models are favored in the combination. The method is designed for regression, using similarities and distances.

Table 3.1, which extends the one from Bellet (2012), reviews the main characteristics of the supervised and semi-supervised metric learning approaches for feature vectors that we studied in this section.

3.3.2 Metric Learning for Time Series

An important proportion of the metrics that perform well on time series are not designed to deal with multivariate time series. One trivial way of adapting them is to weigh all features equally, but this results in degrading performance, as it does not take into account the varying importance of features, nor the possible differences in scale. Metric learning can address exactly this problem by learning the weights of the features and the correlations between them from the available training data. Unfortunately, metric learning for time series has been addressed only by a small number of methods, that we present in this section. These methods are mostly designed for k-NN classification, following the orientation of the time series analysis community.

Sun et al. Sun et al. (2010) apply Average Neighborhood Margin Maximization (ANMM) (Wang & Zhang, 2007) to time series classification for the specific application of physiological data. ANMM is a supervised feature extraction method which aims to learn a projection matrix. For each training sample $\mathbf{x}_i$, the method enlarges its margin to neighbors from

other classes $\mathcal{P}_i$, while keeping the distance to its neighbors from the same class $\mathcal{N}_i$ as small as possible. The Mahalanobis distance they learn optimizes the ratio between the sum of distances over positive, respectively negative pairs:

$$\min_{\mathbf{M}} \frac{\sum_{\mathbf{x}_i \in \mathcal{S}} \sum_{\mathbf{x}_j \mathcal{P}_i} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{\mathbf{x}_i \in \mathcal{S}} \sum_{\mathbf{x}_k \mathcal{N}_i} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k)}.$$

Sun et al. (2010) apply ANMM on features extracted from the temporal data in two ways: either using the first two statistical moments of each feature, or using the wavelet coefficients over temporal fixed-size windows.

Lajugie et al. In (Lajugie et al., 2014), the authors propose to learn a Mahalanobis distance for multivariate time series alignment of audio data. Let $X = (\mathbf{A}, \mathbf{B})$ be a pair of time series of same dimension d, but possibly of different lengths $t_{\mathbf{A}}$ and $t_{\mathbf{B}}$, where $\mathbf{A} \in \mathbb{R}^{t_{\mathbf{A}} \times d}$ and $\mathbf{B} \in \mathbb{R}^{t_{\mathbf{B}} \times d}$. The pairwise cost matrix used for computing the DTW cost is:

$$\mathbf{C}(X, \mathbf{M})_{i,j} = -(\mathbf{a}_i - \mathbf{b}_j)^T \mathbf{M} (\mathbf{a}_i - \mathbf{b}_j),$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_{t_{\mathbf{A}}}$ are the time moments (rows) of $\mathbf{A}$, and similarly for $\mathbf{B}$. Considering groundtruth pairs of the form $\{(X_i, \pi_i^*)\}_{i=1}^n$ known, they learn the Mahalanobis distance in the following way:

$$\min_{\mathbf{M}} \frac{1}{n} \sum_{i=1}^{n} \ell_H(\pi_i^*, \underset{\pi_i \in \mathcal{A}(t_A, t_B)}{\arg\max} \operatorname{tr}(\mathbf{C}(X_i, \mathbf{M})^T \pi_i)) + \lambda \left\| \mathbf{M} \right\|_{\mathcal{F}}^2, \tag{3.15}$$

where λ is the regularization parameter and ℓ_H is the Hamming loss (Hamming, 1950) between two alignments $\pi_1 \in \{0,1\}^{t_A \times t_B}$ and $\pi_2 \in \{0,1\}^{t_A \times t_B}$ encoded as binary matrices:

$$\ell_H(\pi_1, \pi_2) = \left\| \pi_1 - \pi_2 \right\|_{\mathcal{F}}^2.$$

Problem (3.15) is intractable, but the authors solve a large margin surrogate instead. One significant limitation of this approach is that the true alignments are considered a priori known for the audio task, information that is not available in most cases. For this reason, their method cannot be applied in other contexts.

Mei et al. LDMLT (Mei et al., 2015) was designed to learn a Mahalanobis distance for multivariate time series classification from triplet constraints. The loss function over one triplet is computed through the DTW under Mahalanobis distance:

$$\ell(\mathbf{M}, z_i, z_j, z_k) = \rho + \mathrm{DTW}_{\mathbf{M}}(z_i, z_j) - \mathrm{DTW}_{\mathbf{M}}(z_i, z_k),$$

where $\rho > 0$ represents the target margin. The loss function is minimized one triplet at a time under the same LogDet regularizer as ITML. At iteration t, the objective function is thus:

$$\mathbf{M}_{t+1} = \underset{\mathbf{M} \in \mathbb{S}_+^d}{\arg\min}\, D_{ld}(\mathbf{M}, \mathbf{M}_t) + \lambda_t \ell(\mathbf{M}, z_i, z_j, z_k), \tag{3.16}$$

with λ_t controlling the trade-off between satisfying the triplet constraint and keeping the metric matrix close to the one in the previous iteration. Problem (3.16) is solved using a closed-form solution. Experiments are performed for nearest neighbor and SVM classification with good results. However, the loss function they use for the metric learning step is not related to the losses of the classifiers using it afterward.

Chen et al. Model Metric Co-Learning (MMCL) (Chen et al., 2015) uses a nonlinear state space model to learn a metric adapted to univariate and multivariate time series classification. The final representation of a sequence is a linear readout mapping, but the underlying model is a nonlinear dynamic system. The n-dimensional dynamical model is:

$$\begin{aligned} \mathbf{x}(t) &= \tanh(\mathbf{R}\mathbf{x}(t-1) + \mathbf{V}\mathbf{s}(t)), \\ \mathbf{f}(t) &= \mathbf{W}\mathbf{x}(t), \end{aligned} \tag{3.17}$$

where $\mathbf{x}(t) \in \mathbb{R}^n, \mathbf{s}(t) \in \mathbb{R}^m$ and $\mathbf{f}(t) \in \mathbb{R}^m$ are respectively the state vector, the input vector and output at time t; $\mathbf{R} \in \mathbb{R}^{n \times n}$ is a dynamic coupling matrix, $\mathbf{V} \in \mathbb{R}^{m \times n}$ and $\mathbf{W} \in \mathbb{R}^{n \times m}$ are the input and output weight matrices. Equations (3.17) is the State Transition Mapping (STM). MMCL learns a global metric $\mathbf{M}$ under the same formulation as MCML (see Section 3.3.1) to obtain class separation, where the Mahalanobis distance is defined over the weights $\mathbf{W}$

$$d_{\mathbf{M}}(i,j) = (\mathbf{w}_i - \mathbf{w}_j)^T \mathbf{M} (\mathbf{w}_i - \mathbf{w}_j) \tag{3.18}$$

The metric operates over a nonlinear state space model with a low representation cost, expressed as:

$$Q_p(\mathbf{r}, \mathbf{W}) = \sum_{\mathbf{s} \in \mathcal{S}} \sum_{t=1}^{t_{\mathbf{s}}} \|\mathbf{f}(t) - \mathbf{s}(t+1)\|^2 + \eta \|W\|^2 .$$

This equation aims to minimize the difference between actual output $\mathbf{f}(t)$ and desired output $\mathbf{s}(t+1)$. A trade-off parameter λ controls the importance of these costs in the final formulation:

$$\min_{\mathbf{r}, \mathbf{M}} Q_s(\mathbf{r}, \mathbf{M}) + \lambda Q_p(\mathbf{r}, \mathbf{W}),$$

where $\mathbf{r}$ are the parameters of the readout model and $Q_s(\mathbf{r}, \mathbf{M})$ is the loss function from MCML using the distance in Equation (3.18). The problem is solved through alternating between state space model learning and metric learning in the readout model space.

Zhao et al. metricDTW (Zhao et al., 2016) proposes an adaptation of LMNN for Mahalanobis distance learning in the case of univariate time series k-NN classification. The method extracts local time series descriptors, clustering them and learning one metric per pair of clusters. The Mahalanobis distance is computed under DTW alignments computed beforehand. However, the matrices are constrained to be diagonal with the same value all over, which boils down to learning a scalar instead of a metric.

Shen et al. Shen et al. (2016) propose LMNN-DTW, a different adaptation of LMNN to the multivariate time series case. Their objective is to learn a Mahalanobis distance for k-NN classification. Their formulation is mostly similar to LMNN, with the exception that the distance function is replaces with the cost of DTW under metric $\mathbf{M}$:

$$\begin{aligned}\min_{\mathbf{M}\in\mathbb{S}_+^d} (1-\mu) \sum_{(i,j)\in\mathcal{P}} \mathrm{DTW}_{\mathbf{M}}(X_i, X_j) \\ +\mu \sum_{(i,j,l)\in\mathcal{R}} (1-y_{il})[1+\mathrm{DTW}_{\mathbf{M}}(X_i,X_j)-\mathrm{DTW}_{\mathbf{M}}(X_i,X_l)]_+,\end{aligned} \tag{3.19}$$

where $\mathcal{P}$ and $\mathcal{R}$ are respectively the set of positive pairs and the set of relative triplets, both constructed as in LMNN, and μ is a trade-off parameter. The term y_{il} is 1 when X_i and X_l share the same label, and 0 otherwise. The problem with Formulation (3.19) is that the measure of DTW under a metric $\mathbf{M}$ also takes into account the optimal alignment. As the optimal alignment is computed through dynamic programming and changes when the metric is changed, optimizing it is NP-hard. Moreover, this combinatorial factor makes the objective function non-convex and non-differentiable. To overcome this impediment, the authors propose an iterative approach, which first computes the best alignment through DTW under a fixed metric, then solves Problem (3.19) for a fixed alignment, much like standard LMNN. These two steps are repeated until convergence. To try and alleviate the problem of local minima, the authors use random initializations for the metric matrix $\mathbf{M}$. In practice, LMNN-DTW has a higher order of complexity than similar methods (e.g. LDMLT), thus being slower, while achieving inferior performance.

3.4 Conclusion

In this chapter, we have discussed standard metrics for feature vectors and time series. Their limitations justify the important body of work in metric learning aiming to provide task-specific measures. We have given an overview of a significant number of methods for supervised and semi-supervised metric learning for feature vectors and time series. The following points summarize the state of the art methods presented previously:

- Over the past 15 years, metric learning for feature vectors has focused on providing practical solutions for a computationally expensive problem. Currently, state of the art methods perform well and are efficient. However, this focus on practical aspects has been in the detriment of developing a theoretical foundation for metric learning. More precisely, only a small number of studies have been concerned with establishing the consistency of the learned metric on unseen data. Another question that has only rarely been answered is how do the properties of the metric relate to the capacities of the classifier using it, and what can be said about the link between the empirical risk and the true risk.

- A small number of metric learning approaches exists for semi-supervised classification. These methods explore a limited number of ways to incorporate unlabeled information, mostly through adjacency relations. None of these methods come with any theoretical

guarantees over the metric or the classifier.

- Metric learning for time series has been addressed by a relatively small number of studies, mainly because of the additional computational complexity introduced by this type of data. Most of the existing methods are adaptations of existing metric learning algorithms for feature vectors to temporal data. Furthermore, they are often applied on univariate time series, although multivariate temporal data is frequent in applications. The question of how to weigh multiple features across time remains a challenge. Moreover, if for feature vectors recent studies have started developing theoretical guarantees, this is not the case for metrics learned for time series.

- Metrics learned for feature vectors and time series classification are usually associated with the k-NN rule, while other types of classifiers have not extensively been explored.

The contributions in this dissertation aim to address the limitations of the state of the art methods. Chapter 4 is devoted to semi-supervised metric learning for feature vectors. We propose a generic framework that is capable of jointly learning a similarity function and a global linear separator from partially labeled data. We show that this framework can be used with a large number of similarity functions and regularizers. We derive two generalization bounds for our approach, one based on algorithmic robustness, the other one based on Rademacher complexity. Lastly, we compare these two frameworks in terms of results derivation and quality of the bounds. Chapter 5 is dedicated to learning metrics for multivariate time series classification. We propose a parameterized similarity function for time series and a method for learning it. We prove the consistency of our algorithm through arguments of uniform stability. The learned similarity function is used to induce a linear separator with good classification guarantees in the feature space.

Chapter 4

Joint Similarity and Classifier Learning for Feature Vectors

Chapter abstract

In this chapter, we propose a novel, generic framework for performing similarity learning at the same time as learning a global linear classifier from feature vectors. The formulation is capable of leveraging information from unsupervised data additionally to a labeled training set, making it a semi-supervised setting. We show that our framework can be used with a large class of regularizers and many similarity functions. We provide a theoretical analysis of this joint learning formulation through two different frameworks, the algorithmic robustness and the uniform convergence based on the Rademacher complexity, which we further compare. The theoretical results hold for the learned metric and the classifier at the same time. Moreover, they are generic and cover different similarity functions and regularizers without enforcing strong constraints. Experiments conducted on standard datasets show the benefits of our approach over state of the art methods: JSL is efficient and performant.

The content of this chapter is based on the following international publications:

Maria-Irina Nicolae, Éric Gaussier, Amaury Habrard, and Marc Sebban. Joint semi-supervised similarity learning for linear classification. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 594–609, 2015a.

Maria-Irina Nicolae, Marc Sebban, Amaury Habrard, Éric Gaussier, and Massih-Reza Amini. Algorithmic Robustness for Semi-Supervised (ϵ, γ, τ)-Good Metric Learning. In *Proceedings of the 22nd International Conference on Neural Information Processing (ICONIP)*, pages 253–263, 2015b.

Maria-Irina Nicolae, Marc Sebban, Amaury Habrard, Éric Gaussier, and Massih-Reza Amini. Algorithmic Robustness for Learning via (ϵ, γ, τ)-Good Similarity Functions. In *ICLR Workshop*, 2015.

4.1 Introduction

As we have seen in the previous chapter, state of the art in metric learning for feature vectors is dominated by fully supervised methods, most of which are designed for k-NN

classification. These metrics are mostly Mahalanobis distances, constraining the parameter matrix to be PSD and symmetric. However, the learned metrics are in most cases not guaranteed to perform well and lack generalization guarantees.

Balcan et al. (2008b) have proposed a theoretical framework that relates the properties of similarity functions to their performance in learning. This theory is based on an intuitive and practical definition of what makes a good similarity function. Essentially, a similarity K is (ϵ, γ, τ)-good for a given binary classification problem if a proportion of $1 - \epsilon$ examples are on average more similar to *reasonable examples* of the same class than to *reasonable examples* of the opposite class, where a τ proportion of the examples are considered *reasonable*. Under K, the classes should be well separated by a margin of γ. The similarity K is not requested to be a distance, nor positive semi-definite, but can be any bounded similarity function. Given that K has these properties, the (ϵ, γ, τ)-good framework provides generalization guarantees on a linear classifier learned from the similarity. This separator can be learned efficiently using a linear program (LP) and can enforce sparsity due to an L_1 norm constraint.

The first contribution in this chapter is to propose a new, generic similarity and linear classifier learning formulation for feature vectors. Our setting (JSL, for Joint Similarity Learning) can accommodate a large range of similarity functions that are not required to be PSD, and different regularizers. For the examples of similarities that we study, the formulation is convex and can be solved efficiently, potentially leading to sparse solutions. We propose two variants of JSL: one which learns a full matrix, and another which limits the numbers of parameters by only learning a diagonal matrix (JSL-diag), thus being able to obtain a metric from a small amount of data. Optimizing the (ϵ, γ, τ)-goodness of the similarity function preserves the theoretical guarantees from Balcan et al. (2008b) on the classifier in relation to the properties of the similarity. Furthermore, and unlike Bellet et al. (2012), we propose here to jointly learn the metric and the classifier at the same time. This allows learning both the metric and the separator in a semi-supervised way, thus making use of the additional information coming from unlabeled data. To our knowledge, these two learning steps have never been performed jointly in metric learning. Semi-supervision makes JSL appropriate for a practical situation frequent in learning when the annotation of data is expensive, but unlabeled data can be easily obtained. We provide a complete theoretical analysis of our approach using two different frameworks: algorithmic robustness and uniform convergence with Rademacher complexity. We use them to derive two consistency bounds for the joint optimization problem, which we compare and discuss. Lastly, we provide an empirical study on classic datasets and compare our method to different families of supervised and semi-supervised learning algorithms, with or without metric learning.

The rest of this chapter is organized as follows. In Section 4.2, we introduce the theory of (ϵ, γ, τ)-good similarity functions. Section 4.3 presents the formulation of JSL, our joint similarity and classifier learning framework, followed by some examples of similarity functions and regularizers that can be used with it. In Section 4.4, we propose a theoretical analysis of JSL, through the frameworks of algorithmic robustness and Rademacher complexity, leading to the derivation of generalization bounds. Section 4.5 proposes

JSL-diag and its theoretical guarantees for learning a diagonal metric. The experimental evaluation in Section 4.6 proves the functional capacities of the proposed approach, with different similarities and regularizers. We compare JSL against a large number of machine learning and metric learning methods, some fully supervised, and others semi-supervised. Finally, we conclude our contribution in Section 4.7.

4.2 (ϵ, γ, τ)-Good Similarities Framework

Balcan & Blum (2006); Balcan et al. (2008a,b) proposed a new learning theory for similarity functions. The goal of this framework is to generalize over learning with kernels, by relaxing some constraints, while still providing guarantees over the results. The theory of kernels is based on the large margin separation achieved between classes in the (sometimes) implicit and unknown mapping space of the kernel. However, the separation is not visible or evident in the original space of the data, making the design of kernels difficult. Moreover, the positive semi-definiteness constraint on kernels excludes from usage many intuitive similarity functions for the given task. To overcome these limitations, Balcan et al. define a new notion of good similarity function as follows.

Definition 4.1. *(Balcan et al., 2008b) A similarity function K is an (ϵ, γ, τ)-good similarity function for a learning problem P if there exists a (random) indicator function $R(\mathbf{x})$ defining a (probabilistic) set of "reasonable points" such that the following conditions hold:*

1. *A $1-\epsilon$ probability mass of examples $(\mathbf{x}, y)$ satisfy:*

$$\mathbb{E}_{(\mathbf{x}',y')\sim P}\left[yy'K(\mathbf{x},\mathbf{x}')\right] \geq \gamma,$$

2. $\Pr_{\mathbf{x}'}(R(\mathbf{x}')) \geq \tau$.

The first condition in Definition 4.1 can be interpreted as having a $(1-\epsilon)$ proportion of examples $\mathbf{x}$ on average 2γ more similar to random reasonable examples $\mathbf{x}'$ of their own label than to random reasonable examples $\mathbf{x}'$ of the other label. It also expresses the tolerated margin violations in an averaged way: this allows for more flexibility than pair- or triplet-based constraints. The second condition sets the minimum mass of reasonable points one must consider (greater than τ). Notice that no constraint is imposed on the form of the similarity function. The definition covers kernel functions, as well as a large class of non PSD and non symmetric bounded similarities. More importantly, Balcan et al. (2008b) show that good similarity functions under Definition 4.1 can be used to learn a linear separator with good properties, i.e. low true risk.

Consider an (ϵ, γ, τ)-good similarity function K. If the set of reasonable points $R = \{(\mathbf{x}'_1, y'_1), \ldots, (\mathbf{x}'_r, y'_r)\}$ of size r is known, then, by using the empirical version of the expected values in Definition 4.1, the classifier achieving a true risk smaller than ϵ at margin γ is:

$$h(\mathbf{x}) = \operatorname{sgn}\left[\frac{1}{r}\sum_{i=1}^{r} y'_i K(\mathbf{x}, \mathbf{x}'_i)\right].$$

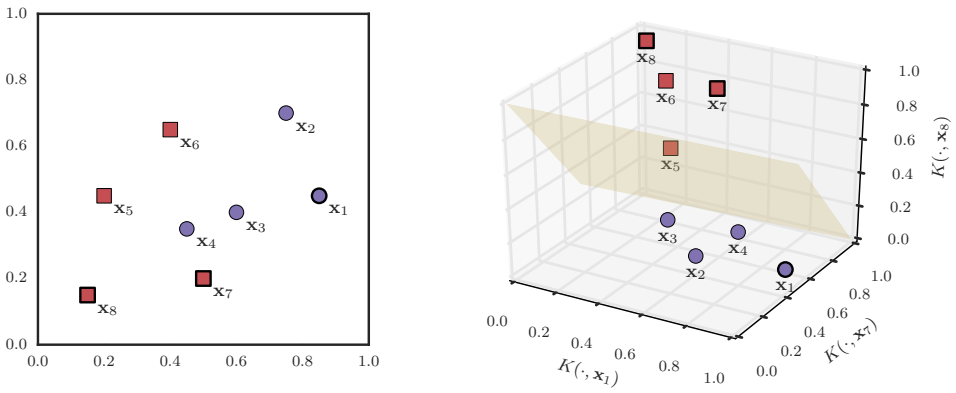


Figure 4.1: Example of (ϵ, γ, τ)-good similarity function. Left side: the original data in the input space is not separable; right side: the similarity projection space, where the separator classifies all points correctly. Reasonable points are outlined in black.

Here, h is a linear classifier in the space of the similarity scores to the reasonable points. This implies that the data is projected through K with respect to the reasonable points using the mapping $\phi : \mathcal{X} \to \mathbb{R}^r$,

$$\phi_i(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}'_i), \quad i \in \{1, \ldots, r\}.$$

An example of input and similarity spaces is shown in Figure 4.1.

In practice, the set of reasonable points is not necessarily known. In this case, a set of landmarks can be sampled from the data to replace them. The number of landmarks needed is proportional to the parameter τ, the probability of a point being reasonable. The landmarks are used in the same way as the reasonable points, that is to construct the feature space. Notice that, in accordance with Definition 4.1, the labels of the landmarks do not have to be known. We will be exploiting this advantage in the contribution that we present in this chapter. Provided that enough landmarks are sampled, there exists with high probability a linear classifier that obtains true error close to ϵ. This is formally expressed in Theorem 4.2.

Theorem 4.2. *(Balcan et al., 2008b) Let K be an (ϵ, γ, τ)-good similarity function for a learning problem P. Let $\mathcal{L} = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_d\}$ be a (potentially unlabeled) sample of $d = \frac{2}{\tau}\left(\log(2/\delta) + 8\frac{\log(2/\delta)}{\gamma^2}\right)$ landmarks drawn from P. Consider the mapping $\phi^{\mathcal{L}} : \mathcal{X} \to \mathbb{R}^d$ defined as follows: $\phi^{\mathcal{L}}(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}'_i), i \in \{1, \ldots, d\}$. Then, with probability $1 - \delta$ over the random sample $\mathcal{L}$, the induced distribution $\phi^{\mathcal{L}}(P)$ in $\mathbb{R}^d$ has a separator error at most $\epsilon + \delta$ relative to L_1 margin at least $\gamma/2$.*

In other words, if K is (ϵ, γ, τ)-good according to Definition 4.1 and enough points are available, there exists a linear separator $\boldsymbol{\alpha}$ with error arbitrarily close to ϵ in the space $\phi^{\mathcal{S}}$.

The previous definition and theorem are expressed with respect to the error and L_1 margin violation. In practice, minimizing the number of violations for the zero-one loss is NP-hard. The authors propose to overcome this limitation by using the hinge loss as a surrogate

function. Making this change leads to a new form of Definition 4.1.

Definition 4.3. *(Balcan et al., 2008b) K is a (ϵ, γ, τ)-good similarity function in hinge loss for a learning problem P if there exists a random indicator function $R(\mathbf{x})$ defining a probabilistic set of "reasonable points" such that the following conditions hold:*

1. *We have*
$$\mathbb{E}_{(\mathbf{x},y)\sim P}\left[[1 - yg(\mathbf{x})/\gamma]_+\right] \leq \epsilon,$$
where $g(\mathbf{x}) = \mathbb{E}_{(\mathbf{x}',y'),R(\mathbf{x}')}\left[y'K(\mathbf{x},\mathbf{x}')|R(\mathbf{x}')\right]$.
2. $\Pr_{\mathbf{x}'}(R(\mathbf{x}')) \geq \tau$.

Under this definition, Theorem 4.2 can be rewritten with respect to the hinge loss:

Theorem 4.4. *(Balcan et al., 2008b) Let K be an (ϵ, γ, τ)-good similarity function in hinge loss for a learning problem P. For any $\epsilon_1 > 0$ and $0 < \delta < \gamma\epsilon_1/4$ let $\mathcal{L} = \{\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_d\}$ be a sample of $d = \frac{2}{\tau}\left(log(2/\delta) + 16\frac{log(2/\delta)}{(\epsilon_1\gamma)^2}\right)$ landmarks drawn from P. Consider the mapping $\phi^{\mathcal{L}} : \mathcal{X} \to \mathbb{R}^d$, $\phi^{\mathcal{L}}_i(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}'_i), i \in \{1, \ldots, d\}$. With probability $1 - \delta$ over the random sample $\mathcal{L}$, the induced distribution $\phi^{\mathcal{L}}(P)$ in $\mathbb{R}^d$, has a separator achieving hinge loss at most $\epsilon + \epsilon_1$ at margin γ.*

One should notice that the transition from the zero-one loss to the hinge loss marginally increases the error of the separator. The procedure for finding the linear separator $\boldsymbol{\alpha} \in \mathbb{R}^d$ that has low true risk involves two steps: first using d_u potentially unlabeled examples as landmarks to construct the feature space, then using a new labeled set of size d_l to estimate $\boldsymbol{\alpha} \in \mathbb{R}^{d_u}$. Given a set of d_u landmarks $\mathcal{L} = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_{d_u}\}$ and a labeled sample of d_l examples $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{d_l}, y_{d_l})\}$, the separator $\boldsymbol{\alpha}$ can be found by solving the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \sum_{i=1}^{d_l}\left[1 - \sum_{j=1}^{d_u}\alpha_j y_i K(\mathbf{x}_i, \mathbf{x}_j)\right]_+ \\ &\text{s.t.} \sum_{j=1}^{d_u}|\alpha_j| \leq 1/\gamma. \end{aligned} \tag{4.1}$$

This formulation can be solved efficiently by linear programming. Furthermore, as it is L_1-constrained, tuning the value of γ will produce a sparse solution. Problem (4.1) can be seen as an equivalent of an L_1 SVM (Zhu et al., 2004). However, some important differences distinguish the two approaches. As seen earlier, K is not required to be symmetric nor PSD, thus generalizing over the notion of kernel. Moreover, the similarity function allows to explicitly create the projection space containing the separator instead of considering an implicit Hilbert space induced by a kernel.

The classification rule based on the similarity K and the separator $\boldsymbol{\alpha}$ takes the following form:

$$y = \operatorname{sgn} \sum_{j=1}^{d_u} \alpha_j K(\mathbf{x}, \mathbf{x}_j). \tag{4.2}$$

The framework of (ϵ, γ, τ)-good similarity functions allows to evaluate the performance that can be expected of a global linear separator depending on how well a similarity function satisfies Definition 4.1. Unfortunately, nothing is said about a potential method to design a good similarity; the function is considered known and fixed. Bellet et al. (2012) have shown how to directly optimize the (ϵ, γ, τ)-goodness of a bilinear similarity function, before plugging it in Equation (4.2) to learn the classifier. Yet the similarity learning step is done in a completely supervised way, while the setting in Balcan et al. (2008b) opens the door to the use of unlabeled data. In the next sections, we present our contribution on learning (ϵ, γ, τ)-good similarity functions from data.

4.3 Joint Similarity and Classifier Learning

In this section, we propose a novel similarity and classifier learning framework based on the theory from Balcan et al. (2008b). We extend this framework to jointly learn the similarity and the separator in a semi-supervised way.

Our goal is to optimize the (ϵ, γ, τ)-goodness of a given similarity function. To this end, we have access to a labeled sample $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{d_l}$ of d_l examples defined over $\mathcal{Z} = \mathcal{X} \times \{-1; +1\}$) coming from an unknown probability distribution P, where $\mathcal{X} \subseteq \mathbb{R}^d$. We are also given a set $\mathcal{L} = \{\mathbf{x}'_j\}_{j=1}^{d_u}$ of d_u unlabeled examples also coming from $\mathcal{X}$. These represent the set of landmarks for building the feature space. Their selection will be discussed into more detail in the experiments. Furthermore, let $K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$ be a generic similarity function, parameterized by the matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$, whose (ϵ, γ, τ)-goodness we aim to optimize.

We assume that $K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') \in [-1; +1]$ and that $||\mathbf{x}||_2 \leq 1$, but all our developments and results can directly be extended to any bounded similarities and datasets. Following the definition of (ϵ, γ, τ)-goodness (Definition 4.3), we want to optimize the empirical goodness of $K_{\mathbf{M}}$ over the training sample with respect to a given landmarks set. The empirical risk of a training instance $z = (\mathbf{x}, y)$ is thus:

$$\ell(\mathbf{M}, \boldsymbol{\alpha}, z, \mathcal{L}) = \left[1 - y \sum_{j=1}^{d_u} \alpha_j K(\mathbf{x}, \mathbf{x}_j) \right]_+ .$$

Our goal here is to find the matrix $\mathbf{M}$ and the global separator $\boldsymbol{\alpha} \in \mathbb{R}^{d_u}$ that minimize the previous empirical loss over the whole sample $\mathcal{S}$, with some guarantees on the generalization error of the associated classifier. To this end, we propose the following regularized formulation based on the joint optimization of the metric and the global separator:

$$\min_{\boldsymbol{\alpha},\mathbf{M}} \quad \sum_{i=1}^{d_l} \Big[1 - y_i \sum_{j=1}^{d_u} \alpha_j K_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)\Big]_+ + \lambda ||\mathbf{M} - \mathbf{R}|| \tag{4.3}$$

$$\text{s.t.} \quad \sum_{j=1}^{d_u} |\alpha_j| \leq 1/\gamma \tag{4.4}$$

$$\|\mathbf{M}\|_{\mathcal{F}} \leq 1, \tag{4.5}$$

where $\lambda > 0$ is a regularization parameter, and $\mathbf{R} \in \mathbb{R}^{d \times d}$ is a fixed matrix such that $\|\mathbf{R}\|_{\mathcal{F}} \leq 1$. The notation $||\cdot||$ refers to a generic matrix norm, for instance L_1, $L_{2,1}$ or Frobenius norms.

The novelty of this formulation is the *joint optimization* over $\mathbf{M}$ and $\boldsymbol{\alpha}$: by solving Problem (4.3), we are learning the metric and the separator at the same time. One of its significant advantages is that it extends the semi-supervised setting from the separator learning step to the metric learning, and the two problems are solved using the same data. This method can naturally be used in situations where one has access to few labeled examples and some unlabeled ones: the labeled examples are used in this case to select the unlabeled examples that will serve to classify new points.

JSL is fundamentally different from standard metric learning approaches presented in Section 3.3.1, which are based on constraints over pairs and triplets of points. The advantage of JSL over these methods is that the constraints on the pairs of points do not need to be satisfied entirely, they only need to be met on average over the set of landmarks. In other words, this formulation is less restrictive than pair or triplet-based settings, and the constraints can be satisfied more easily. Moreover, as the number of landmarks can be much smaller than the size of the training set, generating all the pairs necessary for the computations is less expensive. As the set of landmarks is the same for all the training points, the constraints are global, learning a pair of global similarity and classifier. Constraint (4.4) takes into account the desired margin γ and is the same as in Balcan et al. (2008b). Constraint (4.5) ensures that the learned similarity is bounded. Once $\mathbf{M}$ and $\boldsymbol{\alpha}$ have been learned, the associated binary classifier takes the form given in Equation (4.2).

Regularization and prior knowledge The regularization term $\|\mathbf{M} - \mathbf{R}\|$ serves to limit the complexity of the learned similarity function $K_{\mathbf{M}}$ and can also contain prior knowledge about the form of the metric. The type of norm used and the regularization parameter λ can push more or less strongly the value of $\mathbf{M}$ towards the bias: Frobenius norm will ensure the values in $\mathbf{M}$ are not too far from those in $\mathbf{R}$, while L_1 norm might push some of the entries of $\mathbf{M}$ to match exactly their counterparts in $\mathbf{R}$. The prior knowledge about the task or the form of the metric is encoded in the matrix $\mathbf{R}$, in a way similar to what is proposed in Davis et al. (2007). If the non parameterized version of the similarity considered performs well, then a natural choice for $\mathbf{R}$ is the (rescaled) identity matrix $\frac{1}{d} \cdot \mathbf{I}$. This way, the learned matrix will preserve the good properties of the non parameterized version and will improve it through learning. Another type of information that can be

incorporated in $\mathbf{R}$ is the correlation between features. This can be achieved by setting $\mathbf{R}$ to the inverse of the covariance matrix of the data, as used in the original version of the Mahalanobis distance. Similar information can be encoded concerning the importance of each feature per class, giving more weight to features that are more representative of one of the classes $\{+1; -1\}$. We now propose a new method to capture the importance of each feature for discriminating a certain class. This heuristic is based on the distributions of each feature for each of the two classes through the Kullback–Leibler (KL) divergence. We assume here that each feature $k \in \{1, \ldots, d\}$ follows a Gaussian distribution in each class, with respective empirical means μ_{k+} (class $+1$) and μ_{k-} (class -1) and standard deviations σ_{k+} (class $+1$) and σ_{k-} (class -1). We can compute the empirical value of the KL divergence from a given sample as:

$$D_{KL}^k = \log\left(\frac{\sigma_{k+}}{\sigma_{k-}}\right) + \frac{1}{2}\left(\frac{\sigma_{k+}^2}{\sigma_{k-}^2} - \frac{\sigma_{k-}^2}{\sigma_{k+}^2} + \frac{(\mu_{k-} - \mu_{k+})^2}{\sigma_{k-}^2}\right), \quad 1 \leq k \leq d.$$

and the matrix $\mathbf{R}$ corresponds to $\text{diag}(D_{KL}^1, D_{KL}^2, \cdots, D_{KL}^d)$.

Similarity functions In order to prove the versatility of JSL, we now propose three examples of similarity functions that can be incorporated in our formulation.

Let $K_{\mathbf{M}}^1$ be the bilinear form:

$$K_{\mathbf{M}}^1(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{M} \mathbf{x}'.$$

We also define $K_{\mathbf{M}}^2$:

$$K_{\mathbf{M}}^2(\mathbf{x}, \mathbf{x}') = 1 - (\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}').$$

Similarly, let $K_{\mathbf{M}}^3$ be:

$$K_{\mathbf{M}}^3(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')}{2\sigma^2}\right).$$

$K_{\mathbf{M}}^1$ and $K_{\mathbf{M}}^2$ are linear with respect to their arguments. $K_{\mathbf{M}}^2$ is a straightforward transformation of the Mahalanobis distance into a similarity function. $K_{\mathbf{M}}^3$ resembles the Gaussian kernel and is based on the squared Mahalanobis distance. It introduces nonlinearity in the form of a similarity that decreases fast when the distance between points increases (under the parameters of $\mathbf{M}$). Its additional parameter σ allows to control the width of the "neighborhood" in which the pairs of points are scored high. All three similarities are bounded under the constraints of JSL and have the advantage of keeping Problem (4.3) convex. We will study into more detail the theoretical properties of these functions in the following section, when we show how they impact the generalization bounds that we derive for JSL. Note that we will make use of the first two similarity functions $K_{\mathbf{M}}^1$ and $K_{\mathbf{M}}^2$ in our experiments.

Solving JSL The properties of our framework depend on those of the chosen similarity function and the regularizer. Choosing convex similarity function in $\mathbf{M}$, like in the previous examples, and a convex regularizer are sufficient enough properties to solve Problem (4.3) efficiently. In this case, the formulation of JSL is convex for the metric, as well as for the separator (but not necessarily jointly convex). Solving the problem does not require semi-definite programming, like many metric learning approaches, even when using Mahalanobis-based similarity functions (e.g. $K^2_{\mathbf{M}}$ or $K^3_{\mathbf{M}}$), as the similarity is not PSD.

The hinge loss objective is convex, provided that $K_{\mathbf{M}}$ is convex in $\mathbf{M}$, but it is not differentiable over all the domain. JSL can be solved in this form presented in Equation (4.3) through stochastic approaches. We propose to rewrite its formulation by transforming the sum of hinge losses in constraints and introducing slack variables $\boldsymbol{\xi} \in \mathbb{R}^{d_l}_+$:

$$\begin{aligned} \min_{\boldsymbol{\alpha},\mathbf{M},\boldsymbol{\xi}} \quad & \sum_{i=1}^{d_l} \xi_i + \lambda||\mathbf{M}-\mathbf{R}|| \\ \text{s.t. } & 1 - y_i \sum_{j=1}^{d_u} \alpha_j K_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \leq \xi_i, \quad 1 \leq i \leq d_l \\ & \sum_{j=1}^{d_u} |\alpha_j| \leq 1/\gamma \\ & \|\mathbf{M}\|_{\mathcal{F}} \leq 1. \end{aligned} \tag{4.6}$$

We propose to solve Problem (4.6) by alternating optimization steps over $\mathbf{M}$ and $\boldsymbol{\alpha}$. These steps can be performed efficiently by a standard convex optimization solver (e.g. Mosek (ApS, 2015)). In terms of complexity, this formulation has $d^2 + d_l + d_u$ variables and only $d_l + 2$ constraints, as opposed to standard metric learning approaches, where the number of constraints is proportional to the number of example pairs or triplets, i.e. quadratic or cubic in the number of examples. Notice that the number of landmarks does not affect the size of the problem, as the number of constraints does not depend on the size of $\mathcal{L}$.

4.4 Theoretical Guarantees

In this section, we provide a theoretical analysis of JSL which allows us to establish generalization bounds for our joint similarity learning formulation. The analysis holds for a large class of similarity functions and regularizers. We derive two equivalent results using two different frameworks. In Section 4.4.1, we prove the algorithmic robustness of JSL and derive a PAC generalization bound based on this property. The theoretical analysis in Section 4.4.2 is based on the Rademacher complexity and also allows to provide a generalization bound. We compare these two approaches in Section 4.4.3.

For the purpose of the theoretical discussion to follow, let us rewrite the minimization

Problem (4.3) with a more generalized notation of the loss function:

$$\min_{\boldsymbol{\alpha},\mathbf{M}} \quad \frac{1}{d_l}\sum_{i=1}^{d_l}\ell(\mathbf{M},\boldsymbol{\alpha},z_i,\mathcal{L})+\lambda||\mathbf{M}-\mathbf{R}||, \tag{4.7}$$

$$\text{s.t.} \quad \sum_{j=1}^{d_u}|\alpha_j|\leq 1/\gamma \tag{4.8}$$

$$\|\mathbf{M}\|_{\mathcal{F}}\leq 1. \tag{4.9}$$

Recall that $\ell(\mathbf{M},\boldsymbol{\alpha},z_i,\mathcal{L}))=\left[1-y_i\sum_{j=1}^{d_u}\alpha_j K_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)\right]_+$ is the instantaneous loss estimated at point $z_i=(\mathbf{x}_i,y_i)$. Therefore, the optimization Problem (4.7) under Constraints (4.8) and (4.9) reduces to minimizing the empirical risk

$$R^{\ell}_{\mathcal{S}}(\mathbf{M},\boldsymbol{\alpha})=\frac{1}{d_l}\sum_{i=1}^{d_l}\ell(\mathbf{M},\boldsymbol{\alpha},z_i,\mathcal{L})$$

under regularization over the training set $\mathcal{S}$. Let

$$R^{\ell}_{P}(\mathbf{M},\boldsymbol{\alpha})=\mathbb{E}_{z\sim P}\ell(\mathbf{M},\boldsymbol{\alpha},z,\mathcal{L})$$

be the true risk w.r.t. the unknown distribution P. The target of generalization analysis for joint similarity learning is to bound the difference $R^{\ell}_{P}(\mathbf{M}_{\mathcal{S}},\boldsymbol{\alpha}_{\mathcal{S}})-R^{\ell}_{\mathcal{S}}(\mathbf{M}_{\mathcal{S}},\boldsymbol{\alpha}_{\mathcal{S}})$, where $(\mathbf{M}_{\mathcal{S}},\boldsymbol{\alpha}_{\mathcal{S}})$ is the solution of JSL.

Note that the similarity and the separator are learned under a fixed set of landmarks $\mathcal{L}$. This allows us to define the loss function and the risks with respect to individual examples from the training set, without using pairs of points, as it is usually done in metric learning. For this reason, we are able to derive our generalization bounds based on the standard settings of both algorithmic robustness and Rademacher complexity analysis.

4.4.1 Algorithmic Robustness of JSL

We now present a theoretical analysis of our approach through the framework of algorithmic robustness. Our main result in this section is the derivation of a generalization bound (Theorem 4.9). Recall that, roughly speaking, an algorithm is robust if for any test example z' falling in the same subset as a training example z, the gap between the losses associated with z and z' is bounded. We first prove that JSL is robust, then we derive a PAC generalization bound based on this property. We now introduce the definition of l-lipschitzness, which will allow us to bound the value of the loss function and prove the robustness of JSL.

Definition 4.5 (l-lipschitz continuity)**.** *A similarity function $K_{\mathbf{M}}(\mathbf{x},\mathbf{x}')$ parameterized by a matrix $\mathbf{M}$ is l-lipschitz with respect to its first argument if for any $\mathbf{x}_1,\mathbf{x}_2$, we have:*

$$K_{\mathbf{M}}(\mathbf{x}_1,\mathbf{x}')-K_{\mathbf{M}}(\mathbf{x}_2,\mathbf{x}')\leq l\left\|\mathbf{x}_1-\mathbf{x}_2\right\|.$$

The lipschitz continuity bounds the value of a function through the constant representing the slope of the function between its input points. Intuitively, it means that the values of the function vary in a controlled way over a limited input range. Going back to our examples of similarity functions, we prove their l-lipschitzness hereafter.

Lemma 4.6. *The similarity functions $K^1_{\mathbf{M}}$, $K^2_{\mathbf{M}}$ and $K^3_{\mathbf{M}}$ defined previously have the following lipschitzness properties respectively:*

- $K^1_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$ *is 1-lipschitz w.r.t. its first argument.*
- $K^2_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$ *is 4-lipschitz w.r.t. its first argument.*
- $K^3_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$ *is l-lipschitz w.r.t. its first argument with $l = \frac{2}{\sigma^2}\left(\exp\left(\frac{1}{2\sigma^2}\right) - \exp\left(\frac{-1}{2\sigma^2}\right)\right)$.*

The proof of Lemma 4.6 is provided in Appendix B.1. With this property in mind, we can now prove the algorithmic robustness of JSL in the following theorem.

Theorem 4.7 (Algorithmic robustness of JSL)**.** *Given a partition of $\mathcal{Z}$ into M subsets $\{C_i\}$ such that $z = (\mathbf{x}, y)$ and $z' = (\mathbf{x}', y') \in C_i$ and $y = y'$, and provided that $K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$ is l-lipschitz w.r.t. its first argument, the optimization Problem (4.7) with Constraints (4.8) and (4.9) is $(M, \epsilon(\mathcal{S}))$-robust with $\epsilon(\mathcal{S}) = \frac{1}{\gamma} l\rho$, where $\rho = \sup_{\mathbf{x},\mathbf{x}' \in C_i} ||\mathbf{x} - \mathbf{x}'||$.*

Proof.

$$
\begin{aligned}
\left|\ell(\mathbf{M}, \boldsymbol{\alpha}, z) - \ell(\mathbf{M}, \boldsymbol{\alpha}, z')\right| &= \left| \left[1 - \sum_{j=1}^{d_u} \alpha_j y K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}_j)\right]_+ - \left[1 - \sum_{j=1}^{d_u} \alpha_j y' K_{\mathbf{M}}(\mathbf{x}', \mathbf{x}_j)\right]_+ \right| \\
&\leq \left| \sum_{j=1}^{d_u} \alpha_j y' K_{\mathbf{M}}(\mathbf{x}', \mathbf{x}_j) - \sum_{j=1}^{d_u} \alpha_j y K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}_j) \right| \qquad (4.10) \\
&= \left| \sum_{j=1}^{d_u} \alpha_j (K_{\mathbf{M}}(\mathbf{x}', \mathbf{x}_j) - K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}_j)) \right| \\
&\leq \sum_{j=1}^{d_u} |\alpha_j| \cdot \left| K_{\mathbf{M}}(\mathbf{x}', \mathbf{x}_j) - K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}_j) \right| \qquad (4.11) \\
&\leq \sum_{j=1}^{d_u} |\alpha_j| \cdot l||\mathbf{x} - \mathbf{x}'|| \leq \frac{1}{\gamma} l\rho \qquad (4.12)
\end{aligned}
$$

We obtain Inequality (4.10) from the 1-lipschitzness of the hinge loss; Inequality (4.11) comes from triangle inequality; the first inequality on line (4.12) is due to the l-lipschitzness of $K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}_j)$ w.r.t. its first argument, and the result follows from Condition (4.8). Setting $\rho = \sup_{\mathbf{x},\mathbf{x}' \in C_i} ||\mathbf{x} - \mathbf{x}'||_1$, we get the theorem. □

We now give a PAC generalization bound on the true loss making use of the previous robustness result. We present the following concentration inequality that allows one to capture statistical information coming from the different regions of the partition of $\mathcal{Z}$.

Proposition 4.8. *(van der Vaart & Wellner, 1996) Let* $(|N_1|, \ldots, |N_M|)$ *be an i.i.d. multinomial random variable with parameters* $d_l = \sum_{i=1}^{M} |N_i|$ *and* $(p(C_1), \ldots, p(C_M))$*. By the Bretagnolle-Huber-Carol inequality we have:*

$$\Pr\left\{\sum_{i=1}^{M}\left|\frac{|N_i|}{d_l} - p(C_i)\right| \geq \lambda\right\} \leq 2^M \exp\left(\frac{-d_l\lambda^2}{2}\right),$$

hence with probability at least $1-\delta$*,*

$$\sum_{i=1}^{M}\left|\frac{N_i}{d_l} - p(C_i)\right| \leq \sqrt{\frac{2M\ln 2 + 2\ln(1/\delta)}{d_l}}.$$

We are now able to present the generalization bound for JSL in the following theorem.

Theorem 4.9 (Generalization bound using algorithmic robustness)**.** *Considering that Problem (4.7) is* $(M, \epsilon(\mathcal{S}))$*-robust, that* $(\mathbf{M}_\mathcal{S}, \boldsymbol{\alpha}_\mathcal{S})$ *is its solution learned from sample* $\mathcal{S}$ *and that* $K_\mathbf{M}$ *is* l*-lipschitz w.r.t. to its first argument, for any* $\delta > 0$ *with probability at least* $1-\delta$*, we have:*

$$|R_P^\ell(\mathbf{M}_\mathcal{S}, \boldsymbol{\alpha}_\mathcal{S}) - R_\mathcal{S}^\ell(\mathbf{M}_\mathcal{S}, \boldsymbol{\alpha}_\mathcal{S})| \leq \frac{1}{\gamma} l\rho + B\sqrt{\frac{2M\ln 2 + 2\ln(1/\delta)}{d_l}},$$

where $B = 1 + \frac{1}{\gamma}$ *is an upper bound of the loss* ℓ *and* $\rho = \sup_{\mathbf{x},\mathbf{x}' \in C_i} ||\mathbf{x} - \mathbf{x}'||$*.*

The proof of Theorem 4.9 follows the one described in Xu & Mannor (2010) and is presented in Appendix B.1. Note that the cover radius ρ can be arbitrarily small at the expense of larger values of M. As M appears in the second term, decreasing to 0 when d_l goes to infinity, this bound provides a standard $\mathcal{O}(1/\sqrt{d_l})$ asymptotic convergence.

As one can notice, our main theorem strongly depends on the l-lipschitzness of the similarity function. Plugging the value of l in the bound in Theorem 4.9, we obtain consistency results for Problem (4.7) using a specific similarity. As the gap between empirical and true loss presented in Theorem 4.9 is proportional with the l-lipschitzness of each similarity function, we would like to keep this parameter as small as possible. Concerning our examples of similarity functions, we notice that the generalization bound is tighter for $K^1_\mathbf{M}$ than for $K^2_\mathbf{M}$. The bound for $K^3_\mathbf{M}$ depends on the additional parameter σ, that adjusts the influence of the similarity value w.r.t. the distance to the landmarks.

4.4.2 Uniform Convergence of JSL Using the Rademacher Complexity

In this section, we provide an analysis of JSL based on the uniform convergence framework and the Rademacher complexity, before deriving our main result, a generalization bound for JSL based on these notions (Theorem 4.12).

We start by introducing the definition of what we will consider admissible similarity functions for this analysis. Intuitively, this notion is similar to l-lipschitzness in the sense that both of them bound the values of the similarity function w.r.t. its arguments. If

for l-lipschitzness the bound is computed as a function of the variation of the input, the admissibility takes into account the norms of the examples, as well as the parameters of the metric. Formally, we define admissibility as:

Definition 4.10. *A pairwise similarity function $K_{\mathbf{M}} : \mathcal{X} \times \mathcal{X} \to [-1, 1]$, parameterized by a matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$, is said to be (β, c)-admissible if, for any matrix norm $||\cdot||$, there exist $\beta, c \in \mathbb{R}$ such that $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$,*

$$|K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')| \leq \beta + c \cdot \left\|\mathbf{x}'\mathbf{x}^T\right\| \cdot \|\mathbf{M}\| .$$

We now prove the (β, c) admissibility of our example similarity functions.

Lemma 4.11. *The similarity functions $K^1_{\mathbf{M}}$, $K^2_{\mathbf{M}}$ and $K^3_{\mathbf{M}}$ defined previously have the following admissibility properties respectively:*

- *$K^1_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$ is $(0, 1)$-admissible, that is*

$$|K^1_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')| \leq ||\mathbf{x}'\mathbf{x}^T|| \cdot ||\mathbf{M}||.$$

- *$K^2_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$ is $(1, 4)$-admissible, that is*

$$|K^2_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')| \leq 1 + 4 \cdot ||\mathbf{x}'\mathbf{x}^T|| \cdot ||\mathbf{M}||.$$

- *$K^3_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$ is $\big(\exp(-2/\sigma^2), 0\big)$-admissible, that is*

$$|K^3_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')| \leq \exp\left(-\frac{2}{\sigma^2}\right).$$

The proof of Lemma 4.11 is provided in Appendix B.1.

For any $\mathbf{B}, \mathbf{M} \in \mathbb{R}^{n \times d}$ and any matrix norm $||\cdot||$, its dual norm $||\cdot||_*$ is defined, for any $\mathbf{B}$, by $||\mathbf{B}||_* = \sup_{||\mathbf{M}|| \leq 1} \operatorname{tr}(\mathbf{B}^T\mathbf{M})$, where $\operatorname{tr}(\cdot)$ denotes the trace of a matrix. Denote $X_* = \sup_{\mathbf{x}, \mathbf{x}' \in \mathcal{X}} ||\mathbf{x}'\mathbf{x}^T||_*$. We can now state our generalization bound.

Theorem 4.12 (Generalization bound using Rademacher complexity)**.** *Let $(\mathbf{M}_{\mathcal{S}}, \boldsymbol{\alpha}_{\mathcal{S}})$ be the solution to the joint Problem (4.7) and $K_{\mathbf{M}}$ a (β, c)-admissible similarity function. Then, for any $0 < \delta < 1$, with probability at least $1 - \delta$, the following holds:*

$$|R^{\ell}_P(\mathbf{M}_{\mathcal{S}}, \boldsymbol{\alpha}_{\mathcal{S}}) - R^{\ell}_{\mathcal{S}}(\mathbf{M}_{\mathcal{S}}, \boldsymbol{\alpha}_{\mathcal{S}})| \leq 4\mathfrak{R}_{d_l}\left(\frac{c}{\gamma}\right) + \left(\frac{\beta + cX_*}{\gamma}\right)\sqrt{\frac{2\ln\frac{1}{\delta}}{d_l}},$$

where $X_* = \sup_{\mathbf{x}, \mathbf{x}' \in \mathcal{X}} ||\mathbf{x}'\mathbf{x}^T||_*$.

Theorem 4.12 proves that learning $\mathbf{M}$ and $\boldsymbol{\alpha}$ in a joint manner from a training set minimizes the generalization error, as the latter is bounded by the empirical error of our joint regularized formulation. We discuss into more detail the bound in the next section, where we also compare it against the similar result obtained using the algorithmic robustness of JSL. The proof of Theorem 4.12 makes use of the Rademacher symmetrization

theorem and contraction property (Theorem 4.13 and Lemma 4.14), that we present in the following.

Theorem 4.13. *(Boucheron et al., 2005) Let $\mathfrak{R}_n(\mathcal{F})$ be the Rademacher average over $\mathcal{F}$ as presented in Definition 2.12. We have:*

$$\mathbb{E}\left[\sup_{f\in\mathcal{F}}\left(\mathbb{E}f(\mathcal{S})-\frac{1}{n}\sum_{i=1}^{n}f(z_i)\right)\right]\leq 2\mathfrak{R}_n(\mathcal{F}).$$

Lemma 4.14. *(Ledoux & Talagrand, 1991) Let F be a class of uniformly bounded real-valued functions on (Ω,μ) and $m\in\mathbb{N}$. If for each $i\in\{1,\ldots,m\}$, $\phi_i:\mathbb{R}\to\mathbb{R}$ is a function having a Lipschitz constant c_i, then for any $\{x_i\}_{i\in\mathbb{N}_m}$,*

$$\mathbb{E}_\epsilon\left(\sup_{f\in F}\sum_{i\in\mathbb{N}_m}\epsilon_i\phi_i(f(x_i))\right)\leq 2\mathbb{E}_\epsilon\left(\sup_{f\in F}\sum_{i\in\mathbb{N}_m}c_i\epsilon_i f(x_i)\right).$$

We are now able to present the proof of Theorem 4.12.

Proof of Theorem 4.12. Observe that $R_{\mathcal{S}}(\mathbf{M}_{\mathcal{S}},\boldsymbol{\alpha}_{\mathcal{S}})-R_P(\mathbf{M}_{\mathcal{S}},\boldsymbol{\alpha}_{\mathcal{S}})\leq\sup_{\mathbf{M},\boldsymbol{\alpha}}\left[R_{\mathcal{S}}(\mathbf{M},\boldsymbol{\alpha})-R_P(\mathbf{M},\boldsymbol{\alpha})\right]$, as $(\mathbf{M}_{\mathcal{S}},\boldsymbol{\alpha}_{\mathcal{S}})$ is the solution of Problem 4.7. Also, let $\mathcal{S}=(z_1,\ldots,z_k,\ldots,z_{d_l})$ and $\tilde{\mathcal{S}}=(z_1,\ldots,\tilde{z}_k,\ldots,z_{d_l})$, $1\leq k\leq d_l$, be two close samples that only differ by one instance z_k. Then, we have:

$$\begin{aligned}
&\left|\sup_{\mathbf{M},\boldsymbol{\alpha}}\left[R_{\mathcal{S}}(\mathbf{M},\boldsymbol{\alpha})-R_P(\mathbf{M},\boldsymbol{\alpha})\right]-\sup_{\mathbf{M},\boldsymbol{\alpha}}\left[R_{\tilde{\mathcal{S}}}(\mathbf{M},\boldsymbol{\alpha})-R_P(\mathbf{M},\boldsymbol{\alpha})\right]\right|\\
\leq&\sup_{\mathbf{M},\boldsymbol{\alpha}}\left|R_{\mathcal{S}}(\mathbf{M},\boldsymbol{\alpha})-R_{\tilde{\mathcal{S}}}(\mathbf{M},\boldsymbol{\alpha})\right|\\
=&\frac{1}{d_l}\sup_{\mathbf{M},\boldsymbol{\alpha}}\left|\sum_{z=(\mathbf{x},y)\in\mathcal{S}}\left[1-\sum_{j=1}^{d_u}\alpha_j yK_{\mathbf{M}}(\mathbf{x},\mathbf{x}_j)\right]_+-\sum_{\tilde{z}=(\tilde{\mathbf{x}},\tilde{y})\in\tilde{\mathcal{S}}}\left[1-\sum_{j=1}^{d_u}\alpha_j\tilde{y}K_{\mathbf{M}}(\tilde{\mathbf{x}},\mathbf{x}_j)\right]_+\right|\\
=&\frac{1}{d_l}\sup_{\mathbf{M},\boldsymbol{\alpha}}\left|\left[1-\sum_{j=1}^{d_u}\alpha_j y_kK_{\mathbf{M}}(\mathbf{x}_k,\mathbf{x}_j)\right]_+-\left[1-\sum_{j=1}^{d_u}\alpha_j\tilde{y}_kK_{\mathbf{M}}(\tilde{\mathbf{x}}_k,\mathbf{x}_j)\right]_+\right|\\
=&\frac{1}{d_l}\sup_{\mathbf{M},\boldsymbol{\alpha}}\left|\sum_{j=1}^{d_u}\alpha_j\tilde{y}_kK_{\mathbf{M}}(\tilde{\mathbf{x}}_k,\mathbf{x}_j)-\sum_{j=1}^{d_u}\alpha_j y_kK_{\mathbf{M}}(\mathbf{x}_k,\mathbf{x}_j)\right| \qquad (4.13)\\
\leq&\frac{2}{d_l}\sup_{\mathbf{M},\boldsymbol{\alpha}}\left|\sum_{j=1}^{d_u}\alpha_j y_k^{max}K_{\mathbf{M}}(\mathbf{x}_k^{max},\mathbf{x}_j)\right| \text{ where } z_k^{max}=\operatorname*{arg\,max}_{z=(\mathbf{x},y)\in\{z_k,\tilde{z}_k\}}yK_{\mathbf{M}}(\mathbf{x},\mathbf{x}_j)\\
\leq&\frac{2}{d_l}\sup_{\mathbf{M},\boldsymbol{\alpha}}\left\{\sum_{j=1}^{d_u}|\alpha_j|\cdot|y_k^{max}|\cdot|K_{\mathbf{M}}(\mathbf{x}_k^{max},\mathbf{x}_j)|\right\}\\
\leq&\frac{2}{d_l}\left(\frac{\beta+cX_*}{\gamma}\right) \qquad (4.14)
\end{aligned}$$

Inequality (4.13) comes from the 1-lipschitzness of the hinge loss; Inequality (4.14) comes

from Constraint (4.9), $||\mathbf{M}|| \leq 1$ and the (β, c)-admissibility of $K_{\mathbf{M}}$. Applying McDiarmid's inequality to the term $\sup_{\mathbf{M},\boldsymbol{\alpha}} [R_{\mathcal{S}}(\mathbf{M}, \boldsymbol{\alpha}) - R_P(\mathbf{M}, \boldsymbol{\alpha})]$, with probability $1 - \delta$, we have:

$$\sup_{\mathbf{M},\boldsymbol{\alpha}} [R_{\mathcal{S}}(\mathbf{M}, \boldsymbol{\alpha}) - R_P(\mathbf{M}, \boldsymbol{\alpha})] \leq \mathbb{E}_{\mathcal{S}} \sup_{\mathbf{M},\boldsymbol{\alpha}} [R_{\mathcal{S}}(\mathbf{M}, \boldsymbol{\alpha}) - R_P(\mathbf{M}, \boldsymbol{\alpha})] + \left(\frac{\beta + cX_*}{\gamma}\right) \sqrt{\frac{2 \ln \frac{1}{\delta}}{d_l}}.$$

In order to bound the gap between the true loss and the empirical loss, we now need to bound the expectation term on the right hand side of the above equation.

$$\begin{aligned}
&\mathbb{E}_{\mathcal{S}} \sup_{\mathbf{M},\boldsymbol{\alpha}} [R_{\mathcal{S}}(\mathbf{M}, \boldsymbol{\alpha}) - R_P(\mathbf{M}, \boldsymbol{\alpha})] \\
=&\mathbb{E}_{\mathcal{S}} \sup_{\mathbf{M},\boldsymbol{\alpha}} \left\{ \frac{1}{d_l} \sum_{i=1}^{d_l} \left[1 - \sum_{j=1}^{d_u} \alpha_j y_i K_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \right]_+ - R_P(\mathbf{M}, \boldsymbol{\alpha}) \right\} \\
\leq&2\mathbb{E}_{\mathcal{S},\sigma} \sup_{\mathbf{M},\boldsymbol{\alpha}} \left\{ \frac{1}{d_l} \sum_{i=1}^{d_l} \sigma_i \left[1 - \sum_{j=1}^{d_u} \alpha_j y_i K_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \right]_+ \right\} \qquad (4.15) \\
\leq&4\mathbb{E}_{\mathcal{S},\sigma} \sup_{\mathbf{M},\boldsymbol{\alpha}} \left| \frac{1}{d_l} \sum_{i=1}^{d_l} \sigma_i y_i \sum_{j=1}^{d_u} \alpha_j K_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \right| \qquad (4.16) \\
\leq&4 \left(\frac{c}{\gamma}\right) \mathbb{E}_{\mathcal{S},\sigma} \sup_{\tilde{\mathbf{x}}} \left\| \frac{1}{d_l} \sum_{i=1}^{d_l} \sigma_i y_i \mathbf{x}_i \tilde{\mathbf{x}}^T \right\|_* = 4\mathfrak{R}_{d_l} \left(\frac{c}{\gamma}\right). \qquad (4.17)
\end{aligned}$$

We obtain Inequality (4.15) by applying Theorem 4.13, while Inequality (4.16) comes from the use of Lemma 4.14. The Inequality on line (4.17) makes use of the (β, c)-admissibility of the similarity function $K_{\mathbf{M}}$ (Definition 4.10). Combining Inequalities (4.14) and (4.17) completes the proof of the theorem. □

4.4.3 Discussion

We will now comment on the generalization bounds obtained for JSL (Theorems 4.9 and 4.12) and the two methods we have used to derive them.

The bound derived in Theorem 4.9 using the algorithmic robustness of JSL holds for all l-lipschitz similarity functions and all regularization norms. The latter characteristic comes from the fact that algorithmic robustness does not take into account at all the regularizer, facilitating the development of generic frameworks. Moreover, almost no algorithm-specific argument is needed to derive this type of bounds. This can also be seen as a downside, as the lack of information about how the hypothesis is learned yields very general results, producing loose consistency bounds. For the same reason, algorithmic robustness bounds are mostly similar even from one method to another. On the other hand, as robustness is based on covering numbers, the bound in Theorem 4.9 is parameterized by the number of parts in the partitioning of the space M and the size of each part ρ. As we have shown, there is a trade-off between these two values, both of which make the bound looser. In

practice, these values are sometimes hard to estimate.

The bound derived in Theorem 4.12 depends on the (β, c)-admissibility of the similarity function, which is in some sort equivalent to the l-lipschitz constraint for algorithmic robustness. As we have seen in Section 2.7.2.2, the Rademacher complexity, unlike the VC dimension, is data-dependent and can integrate information about the task. Moreover, the Rademacher complexity of a given task can be estimated from a single training sample $\mathcal{S}$. Usually, Rademacher complexity-based bounds implicitly integrate the regularizer, as it is common practice to bound the matrix $\mathbf{M}$ using this term. This is not the case for JSL, as $\|\mathbf{M}\|_{\mathcal{F}}$ is bounded by a constant, making our bound independent from $\mathbf{R}$ and the chosen matrix norm.

When comparing the two generalization bounds derived for JSL, we notice that they both have the same order of convergence, that is $\mathcal{O}(\frac{1}{\sqrt{d_l}})$ w.r.t. the size of the training set, which is the standard rate for PAC generalization bounds. On the other hand, none of the bounds manifests any dependence on the size of the landmarks set. Moreover, they are also independent of the dimensionality of the data. This property is due to the fact that $\|\mathbf{M}\|_{\mathcal{F}}$ is bounded by a constant. For the same reason, both bounds are agnostic to the regularization, which allows them to cover different types of norms for the term $\|\mathbf{M} - \mathbf{R}\|$, but makes them more loose as a downside. In both cases, the gap between the empirical and the true risk is inversely proportional to the margin γ, implying that the larger the margin (i.e. the separation between classes), the better the empirical risk estimates the true error. An important difference between the bounds resides in their respective first terms. Considering an infinite sample ($d_l \to \infty$), both terms of the Rademacher complexity-based bound go to zero (to see this, recall Definition 2.11 for the empirical Rademacher complexity). This is not the case for the bound based on algorithmic robustness: its first term is constant with respect to the size of the sample, making in our case the Rademacher complexity bound more informative.

To conclude, the two consistency bounds we have presented for JSL are mostly equivalent, with the same rate of convergence and dependence on similar parameters. Analyses based on Rademacher complexity can yield tighter bounds, as they integrate more information specific to the task, but are in practice harder to derive than their equivalent based on algorithmic robustness.

4.5 Learning a Diagonal Metric

In this section, we introduce a modified version of JSL which allows to learn a similarity function parameterized by a diagonal matrix at the same time as a linear classifier. As JSL is adapted for the case of small amounts of labeled information, the number of available examples is not always sufficient for learning the d^2 parameters of a full matrix. We thus propose JSL-diag, where the number of parameters to be learned for the similarity function is d. JSL-diag takes the following form:

$$\begin{aligned} \min_{\boldsymbol{\alpha},\mathbf{M}} \quad & \sum_{i=1}^{d_l} \left[1 - \sum_{j=1}^{d_u} \alpha_j y_i K_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)\right]_+ + \lambda \left\|\mathbf{M} - \mathbf{R}\right\| \\ \text{s.t.} \quad & \sum_{j=1}^{d_u} |\alpha_j| \leq 1/\gamma \\ & \mathbf{M} \text{ diagonal}, \quad |M_{kk}| \leq 1, \quad 1 \leq k \leq d, \end{aligned} \tag{4.18}$$

where $\lambda > 0$ is a regularization parameter, and $\mathbf{R} \in \mathbb{R}^{d \times d}$ is now a fixed diagonal matrix such that $\|\mathbf{R}\| \leq d$. The same heuristics for choosing $\mathbf{R}$ and the same similarity functions can be use for JSL-diag as for JSL. The semantics behind setting the off-diagonal values of $\mathbf{M}$ to zero is that the similarity function will not take into consideration the correlation between features. In practice, it has been shown that the change in performance is marginal with respect to the case of a full matrix (Qamar & Gaussier, 2009a). When introducing slack variables, the diagonal form of $\mathbf{M}$ reduces the number of variables of the problem from $d^2 + d_l + d_u$ to only $d + d_l + d_u$, thus making it linear in the parameters. The constraint however does not change the number of constraints, which remains $d_l + 2$. JSL-diag is solved similarly to JSL, through alternating optimization steps over $\mathbf{M}$ and $\boldsymbol{\alpha}$. We will use JSL-diag in the experimental section.

From a theoretical perspective, JSL-diag has the same properties as JSL, thus making it possible to derive similar generalization bounds. The only distinction comes from the constraint over $\mathbf{M}$ changing from $\|\mathbf{M}\|_{\mathcal{F}} \leq 1$ in the case of JSL to $|M_{kk}| \leq 1, 1 \leq k \leq d$ for JSL-diag. The bound derived using algorithmic robustness Theorem 4.9 is the same for JSL and JSL-diag, as it is based on the partitioning of the space. On the other hand, we can rewrite Theorem 4.12 for JSL-diag as follows. Note that the proofs are the same as for JSL; we will thus omit them.

Theorem 4.15. *Let* $(\mathbf{M}_{\mathcal{S}}, \boldsymbol{\alpha}_{\mathcal{S}})$ *be the solution to the joint problem* (4.18) *and* $K_{\mathbf{M}}$ *a* (β, c)*-admissible similarity function. Then, for any* $0 < \delta < 1$*, with probability at least* $1 - \delta$*, the following holds:*

$$|R_P^{\ell}(\mathbf{M}_{\mathcal{S}}, \boldsymbol{\alpha}_{\mathcal{S}}) - R_{\mathcal{S}}^{\ell}(\mathbf{M}_{\mathcal{S}}, \boldsymbol{\alpha}_{\mathcal{S}})| \leq 4\mathfrak{R}_{d_l}\left(\frac{cd}{\gamma}\right) + \left(\frac{\beta + cX_* d}{\gamma}\right)\sqrt{\frac{2\ln\frac{1}{\delta}}{d_l}},$$

where $X_* = \sup_{\mathbf{x},\mathbf{x}' \in \mathcal{X}} \left\|\mathbf{x}'\mathbf{x}^T\right\|_*$.

Changing the constraint on the bound of the norm of $\mathbf{M}$ makes the generalization bound in Theorem 4.15 less tight. Notice that the norm of $\mathbf{M}$ is now bounded by d, not 1 like in the case of JSL. As a consequence, the consistency bound for JSL-diag depends linearly on d.

4.6 Experimental Validation

The state of the art in metric learning is dominated by algorithms designed to work in a purely supervised setting. Furthermore, most of them optimize a metric adapted to k-NN classification (e.g. LMNN, ITML), while our work is designed for finding a global linear

Table 4.1: Properties of the datasets used in the experimental study.

	Balance	Ionosphere	Iris	Liver	Pima	Sonar	Wine
# Instances	625	351	150	345	768	208	178
# Dimensions	4	34	4	6	8	60	13
# Classes	3	2	3	2	2	2	3

separator. For these reasons, it is difficult to propose a totally fair comparative study. In this section, we first evaluate the effectiveness of JSL-diag with different settings. Secondly, we extensively compare it with state-of-the-art algorithms from different categories (supervised, k-NN oriented). Lastly, we study the impact of the quantity of available labeled data on our method. We conduct the experimental study on 7 classic datasets taken from the UCI Machine Learning Repository (Lichman, 2013), both binary and multi-class. Their characteristics are presented in Table 4.1. These datasets are widely used for metric learning evaluation.

4.6.1 Experimental setting

Compared methods In order to provide a comparison as complete as possible, we propose to study two main families of approaches[1]:

1. *Linear classifiers* – in this family, we consider the following methods:
   - BBS, corresponding to Problem (4.1) and discussed before;
   - SLLC (Bellet et al., 2012), an extension of BBS in which a similarity is learned prior to be used in the BBS framework;
   - JSL-diag, the joint learning framework proposed in this study learning a diagonal similarity;
   - Linear SVM with L_2 regularization, which is the standard approach for linear classification.
2. *Nearest neighbor approaches* – in this family, we consider the methods:
   - Standard 3-nearest neighbor classifier (3-NN) based on the Euclidean distance;
   - ITML (Davis et al., 2007), which learns a Mahalanobis distance that is used here in 3-NN classification;
   - LMNN with a full matrix and LMNN with a diagonal matrix (LMNN-diag) (Weinberger & Saul, 2008, 2009), also learning a Mahalanobis distance used here in 3-NN classification;
   - LRML (Hoi et al., 2008, 2010); LRML also learns a Mahalanobis distance used in 3-NN classifier, but in a semi-supervised setting. This method has the closest

[1]For all the methods, we used the code provided by the authors.

setting to JSL (even though one is learning a linear separator and the other only a distance).

All classifiers are used in their binary version, in a one-vs-all setting when the number of classes is greater than two. BBS, SLLC and JSL rely on the same classifier from Equation (4.2), even though learned in different ways. We solve BBS and JSL using projected gradient descent.

Data processing and parameter settings All features are centered around zero and scaled to ensure $||\mathbf{x}||_2 \leq 1$, as this constraint is necessary for some of the algorithms. We randomly choose 15% of the data for validation purposes, and another 15% as a test set. The training set and the unlabeled data are chosen from the remaining 70% of examples not employed in the previous sets. In order to illustrate the classification using a restricted quantity of labeled data, the number of labeled points is limited to 5, 10 or 20 examples per class, as this is usually a reasonable minimum amount of annotation to rely on. The number of landmarks is either set to 15 points or to all the points in the training set (in which case their label is not taken into account). These two settings correspond to two practical scenarios: one in which a relatively small amount of unlabeled data is available, and one in which a large amount of unlabeled data is available. When only 15 unlabeled points are considered, they are chosen from the training set as the nearest neighbor of the 15 centroids obtained by applying K-means++ clustering with $k = 15$. All of the experimental results are averaged over 10 runs, for which we compute a 95% confidence interval. We tune the following parameters by cross-validation: $\gamma, \lambda \in \{10^{-4}, \ldots, 10^{-1}\}$ for BBS and JSL (λ only for the second), $\lambda_{ITML} \in \{10^{-4}, \ldots, 10^{4}\}$, choosing the value yielding the best accuracy. For SLLC, we tune $\gamma, \beta \in \{10^{-7}, \ldots, 10^{-2}\}$, $\lambda \in \{10^{-3}, \ldots, 10^{2}\}$, as done by the authors, while for LRML we consider $\gamma_s, \gamma_d, \gamma_i \in \{10^{-2}, \ldots, 10^{2}\}$. For LMNN, we set $\mu = 0.5$, as done in Weinberger & Saul (2009).

4.6.2 Experimental results

Analysis of JSL We first study here the behavior of the proposed joint learning framework w.r.t. different families of similarities and regularization functions (choice of $\mathbf{R}$ and $||\cdot||$). In particular, we consider two types of similarity measures: bilinear (cosine-like) similarities of the form $K^1_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{M}\mathbf{x}'$ and similarities derived from the Mahalanobis distance $K^2_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = 1 - (\mathbf{x} - \mathbf{x}')^T\mathbf{M}(\mathbf{x} - \mathbf{x}')$. For the regularization term, $\mathbf{R}$ is either set to the identity matrix (JSL-$\mathbf{I}$), or to the approximation of the Kullback–Leibler divergence (JSL-KL) discussed in Section 4.3. As mentioned above, these two settings correspond to different prior knowledge one can have on the problem. In both cases, we consider L_1 and L_2 regularization norms. We thus obtain 8 settings, that we compare in the situation where few labeled points are available (5 points per class), using a small amount (15 instances) of unlabeled data or a large amount (the whole training set) of unlabeled data. The results of the comparisons are reported in Tables 4.2 and 4.3.

As one can note from Table 4.2, when only 15 points are used as landmarks, $K^2_{\mathbf{M}}$ obtains better results in almost all of the cases, the difference being more important on Iris, Pima

Table 4.2: Classification accuracy (%) of JSL-diag with confidence interval at 95%, 5 labeled points per class, 15 unlabeled landmarks.

Sim.	Reg.	Balance	Ionosphere	Iris	Liver	Pima	Sonar	Wine
$K^1_{\mathbf{M}}$	**I**-L_1	85.2±3.0	85.6±2.4	76.8±3.2	63.3±6.2	71.0±4.1	72.9±3.6	**91.9**±4.2
	I-L_2	85.1±2.9	85.6±2.6	76.8±3.2	63.1±6.3	71.0±4.0	73.2±3.8	91.2±4.5
	KL-L_1	84.9±2.9	85.0±2.6	77.3±2.7	63.9±5.5	71.0±4.0	72.9±3.6	90.8±4.7
	KL-L_2	85.2±3.0	85.8±3.3	76.8±3.2	62.9±6.4	71.3±4.3	74.2±3.8	90.0±5.4
$K^2_{\mathbf{M}}$	**I**-L_1	**87.2**±2.9	**87.7**±2.6	78.6±4.6	64.7±5.6	75.1±3.5	73.9±5.7	80.8±9.5
	I-L_2	86.8±3.0	**87.7**±2.8	75.9±5.7	64.3±5.4	**75.6**±3.6	74.8±5.8	80.8±8.6
	KL-L_1	**87.2**±2.9	87.3±2.4	78.6±4.6	62.9±5.6	75.0±3.7	75.5±6.2	79.6±11.8
	KL-L_2	**87.1**±2.7	85.8±3.3	**79.1**±5.4	**64.9**±5.9	**75.6**±3.4	**77.1**±5.2	79.6±9.7

Table 4.3: Classification accuracy (%) of JSL-diag with confidence interval at 95%, all points used as landmarks.

Sim.	Reg.	Balance	Ionosphere	Iris	Liver	Pima	Sonar	Wine
$K^1_{\mathbf{M}}$	**I**-L_1	85.8±2.9	88.8±2.5	74.5±3.1	65.5±4.5	71.4±3.8	70.3±6.6	85.8±5.0
	I-L_2	85.8±2.9	87.7±2.7	74.5±3.5	64.7±5.5	71.7±4.1	68.7±6.7	84.6±5.5
	KL-L_1	85.6±3.1	87.9±3.4	75.0±3.5	65.3±4.9	71.6±4.2	70.3±6.8	85.4±5.3
	KL-L_2	85.1±3.1	88.5±3.7	**75.9**±3.4	65.1±4.8	72.1±4.2	71.9±6.7	**86.5**±6.0
$K^2_{\mathbf{M}}$	**I**-L_1	85.9±2.3	90.4±2.2	71.8±6.1	67.3±3.5	73.1±3.5	72.9±4.2	81.5±8.4
	I-L_2	**86.2**±2.5	**90.6**±2.2	73.2±6.6	**68.6**±3.3	73.3±3.2	**73.2**±4.2	82.7±9.0
	KL-L_1	85.8±2.6	89.4±2.0	72.7±5.5	67.5±3.8	**73.8**±3.5	71.0±4.1	80.0±7.4
	KL-L_2	85.9±2.4	89.6±2.2	74.5±6.2	68.4±3.6	73.1±3.8	72.3±4.8	80.0±11.5

and Sonar. The noticeable exception to this better behavior of $K^2_{\mathbf{M}}$ is Wine, for which cosine-like similarities outperform Mahalanobis-based similarities by more than 10 points. A similar result was also presented in Qamar et al. (2008). The difference between the use of the L_1 or L_2 norms is not as marked, and there is no strong preference for one or the other, even though the L_2 norm leads to slightly better results in average than the L_1 norm. Regarding the regularization matrix $\mathbf{R}$, again, the difference is not strongly marked, except maybe on Sonar. In average, regularizing through the Kullback-Leibler divergence leads to slightly better results than regularizing through the identity matrix.

When all points are used as landmarks (Table 4.3), similar conclusions can be drawn regarding the similarity functions and the norms used. However, in that case, the regularization based on the identity matrix yields better results than the one based on the KL divergence. It is important to note also that the overall results are in general lower than the ones obtained when only 15 points are used as landmarks. We attribute this effect to the fact that one needs to learn more parameters (via $\boldsymbol{\alpha}$), whereas the amount of available labeled data is the same.

From the above analysis, we focus now on two JSL-diag based methods: JSL-15 with $K^2_{\mathbf{M}}$,

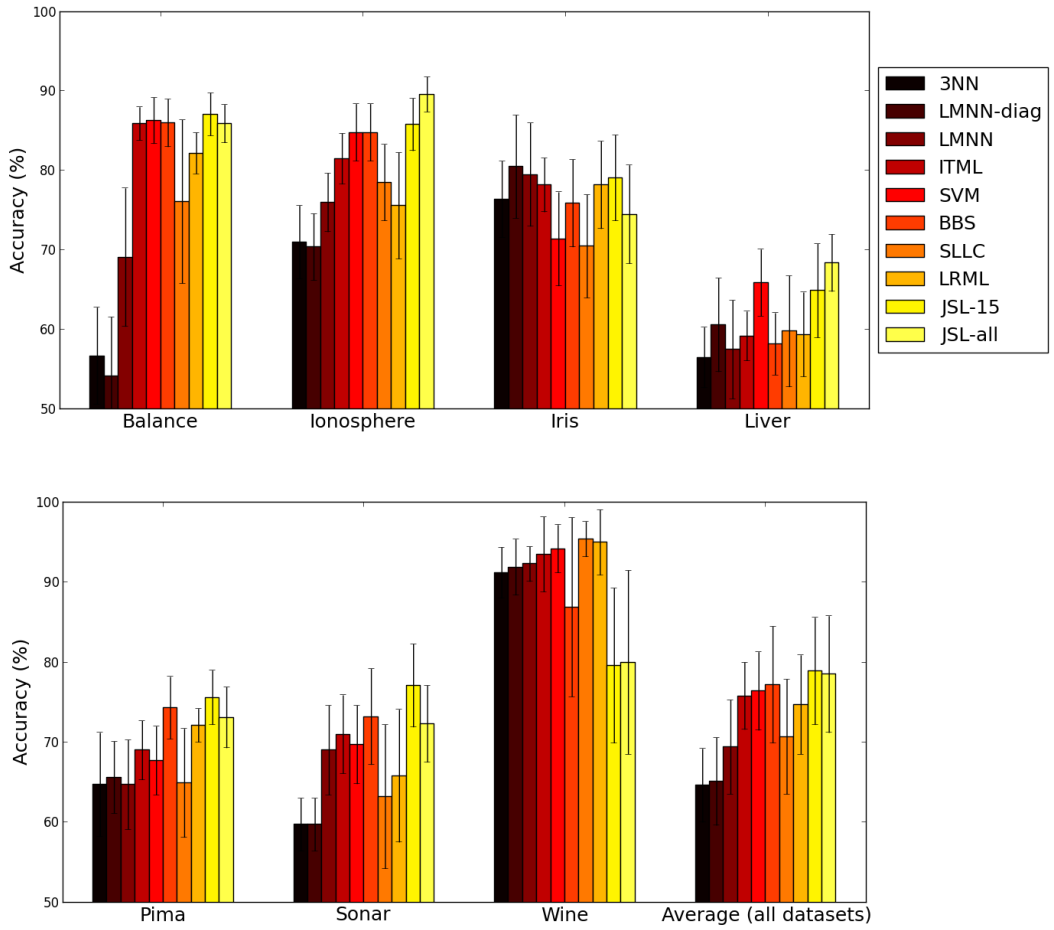


Figure 4.2: Classification accuracy (%) with confidence interval at 95% with 5 labeled points per class.

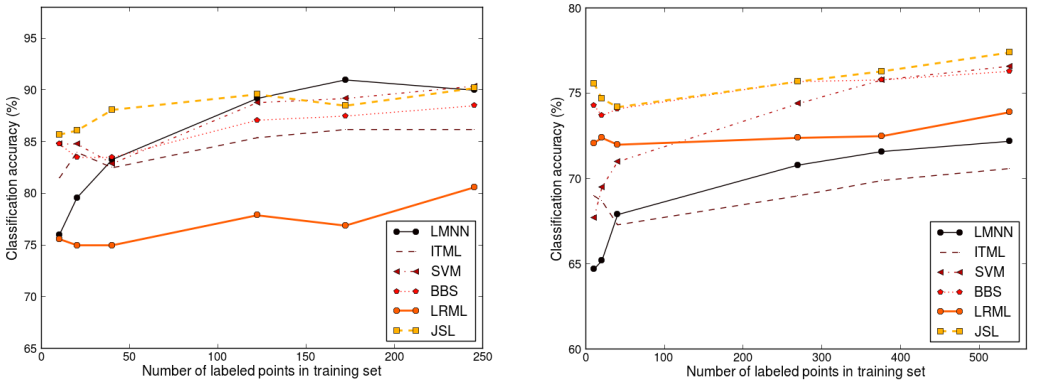
L_2 norm and $\mathbf{R} = KL$ when 15 points are used as landmarks, and JSL-all with $K^2_{\mathbf{M}}$, L_2 norm and $\mathbf{R} = \mathbf{I}$ when all the points are used as landmarks.

Comparison of the different methods We now study the performance of our method, compared to state-of-the-art algorithms. For this, we consider JSL-15 and JSL-all with 5, 10, respectively 20 labeled examples per class. As our methods are tested using the similarity based on the Mahalanobis distance, we use the Euclidean distance for BBS to ensure fairness.

Figure 4.2 presents the average accuracy per dataset obtained with 5 labeled points per class. In this setting, JSL outperforms the other algorithms on 5 out of 7 datasets and has similar performances on one other. The exception is the Wine dataset, where none of the JSL settings yields competitive results. As stated before, this is easily explained by the fact cosine-similarities are more adapted for this dataset. Even though JSL-15 and JSL-all perform the same when averaged over all datasets, the difference between them is marked on some datasets: JSL-15 is considerably better on Iris and Sonar, while JSL-all significantly outperforms JSL-15 on Ionosphere and Liver. Averaged over all datasets (Table 4.4), JSL obtains the best performance in all configurations with a limited amount

Table 4.4: Average classification accuracy (%) over all datasets with confidence interval at 95%.

Method	5 pts./cl.	10 pts./cl.	20 pts./cl.
3-NN	64.6±4.6	68.5±5.4	70.4±5.0
LMNN-diag	65.1±5.5	68.2±5.6	71.5±5.2
LMNN	69.4±5.9	70.9±5.3	73.2±5.2
ITML	75.8±4.2	76.5±4.5	76.3±4.8
SVM	76.4±4.9	76.2±7.0	77.7±6.4
BBS	77.2±7.3	77.0±6.2	77.3±6.3
SLLC	70.5±7.2	75.9±4.5	75.8±4.8
LRML	74.7±6.2	75.3±5.9	75.8±5.2
JSL-15	**78.9**±6.7	**77.6**±5.5	77.7±6.4
JSL-all	78.2±7.3	76.6±5.8	**78.4**±6.7



(a) Ionosphere (b) Pima

Figure 4.3: Classification accuracy w.r.t. the number of labeled points with 15 landmarks.

of labeled data, which is particularly the setting that our method is designed for. The values in bold are significantly better than the rest of their respective columns, confirmed by a one-sided Student t-test for paired samples with a significance level of 5%.

Impact of the amount of labeled data As an illustration of the methods' behavior when the level of supervision varies, Figure 4.3 presents the accuracies on two representative datasets, Ionosphere and Pima, with an increasing number of labeled examples. In both cases, the best results are obtained by JSL (and more precisely JSL-15) when less than 50% of the training set is used. This is in agreement with the results reported in Table 4.4. The results of JSL are furthermore comparable only to BBS for the Pima dataset. Lastly, the accuracy of JSL improves slightly when adding more labeled data, and the results on the whole training set are competitive w.r.t. the other algorithms.

4.7 Conclusion

In this chapter, we have studied the problem of learning similarities in the situation where few labeled (and potentially few unlabeled) examples are available. To do so, we have developed a semi-supervised framework extending the (ϵ, γ, τ)-good of Balcan et al. (2008b), in which the similarity function and the classifier are learned at the same time. To our knowledge, this is the first time that such a framework is provided. The joint learning of the similarity and the classifier enables one to benefit from unlabeled data for both the similarity and the classifier. Our framework is generic and can be used with many different similarity functions and regularizers, some of which are able to provide sparsity for the learned metric in addition to that of the classifier. As JSL optimizes the (ϵ, γ, τ)-goodness of the learned similarity, it also benefits from the guarantees of this framework, i.e. the bound on the true error of the linear classifier. We have also showed that the proposed method was theoretically well-founded through the analyses based on algorithmic robustness and Rademacher complexity, each resulting in a bound on the generalization error of the learned parameters. Lastly, the experiments we have conducted on standard metric learning datasets show that our approach is indeed well-suited for learning a diagonal metric from few labeled data, and outperforms state-of-the-art metric learning approaches in that situation.

Future work could cover a kernelized version of our technique to learn more efficient similarities and classifiers in a nonlinear feature space, as well as learning local metrics and a combination of them in a coherent framework. One could also consider introducing nonlinearity in JSL through the choice of a nonlinear similarity function. In the case of JSL, we have used the classic algorithm K-means to choose the landmarks that are used for the construction of the features space. An interesting line of research could be to analyze multiples heuristics for landmarks selection, as criteria different from data density might yield better performance.

Chapter 5

Learning Similarities for Time Series Classification

Chapter abstract

Dynamic time warping (DTW) is the most well-known algorithm for measuring the similarity between two time series by finding the best alignment between them. Unfortunately, as we have seen in Chapter 3, not much research effort has been put into adapting it to multivariate time series, and even less into improving it by learning. In this chapter, we propose a novel method for learning similarities based on DTW, in order to improve time series classification. In this contribution also we makes use of the (ϵ, γ, τ)-good similarities learning framework (Balcan et al., 2008b), providing guarantees on the performance of a linear classifier built from the learned metric. Proving that our method has uniform stability allows us to derive the first consistency bound for a metric learned from temporal data. We perform experiments on real-world multivariate time series datasets, comparing our method to state of the art approaches. The experimental study shows that the proposed approach is efficient, while yielding sparse classifiers.

The material of this chapter is based on the following technical report:

Maria-Irina Nicolae, Éric Gaussier, Amaury Habrard, and Marc Sebban. Similarity Learning for Time Series Classification. Technical report, University of Saint-Etienne, 2016. arXiv:1610.04783. To be submitted to the journal track of ECML/PKDD 2017 and MLJ.

5.1 Introduction

Dynamic time warping has shown good results for time series tasks, receiving important attention from the research community. As presented in Section 3.3.2, metric learning for time series in general, and for the multivariate case in particular, has only been explored by a limited number of studies. The vast majority of methods are constructed from local constraints and designed for k-NN classification, following the orientation of the time series analysis community. The question remains if classifiers and metrics inferred from global constraints (e.g. linear) can be effective on time series. Often learning a Mahalanobis distance, existing methods impose a PSD constraint which is computationally expensive

and could be avoided by choosing a different class of functions. Moreover, the subject of learning custom metrics for time series has been addressed by adapting standard metric learning solutions to incorporate the time dimension (e.g. LMNN (Weinberger & Saul, 2009), ITML (Davis et al., 2007)). This often implies that the optimal DTW alignment has to be computed for all pairs of points, but considering the quadratic complexity of this procedure, the final computational cost is elevated. Additionally, the theoretical foundation of metric learning for time series classification has never been studied.

In this chapter, we address the limitations of state of the art methods by introducing Similarity Learning for Time Series (SLTS). We place ourselves once more in the framework of (ϵ, γ, τ)-good similarity functions. Our approach is to consider the optimal DTW alignment under a standard metric, but only w.r.t. a set of landmarks of limited size. The formulation we propose is convex and directly optimizes the (ϵ, γ, τ)-goodness of a bilinear similarity based on the optimal alignments, thus being able to guarantee its performance in linear classification. Our setting is not limited to the similarity function that we propose, but can be adapted to other forms of metrics. We provide a comprehensive theoretical analysis of SLTS based on the uniform stability framework. By only being able to incorporate strongly convex regularizers, the uniform stability is less general than algorithmic robustness or Rademacher complexity, but allows the derivation of tighter bounds. This is necessary for SLTS, as the method deals with complex data, which has an additional time dimension. The theoretical study in this chapter results in the derivation of a generalization bound for the learned metric. To our knowledge, SLTS is the first metric learning approach for multivariate time series coming with theoretical guarantees. The experiments we perform cover multiple aspects. We show examples of visualizations of the similarity space to prove that the learned metric has the capacity to discriminate classes. The comparison against state of the art approaches proves that SLTS is efficient and obtains good performance. We additionally study and test diverse heuristics for choosing the set of landmarks on which the construction of the similarity space is based.

Note that an approach similar to SLTS has been proposed in Bellet et al. (2011) for the problem of learning a similarity for sequence alignment. More precisely, their purpose is to learn the cost matrix necessary for computing the edit distance, i.e. the script of operations which transform a string into another. Computing the DTW for time series is equivalent to determining the Levenshtein distance (Levenshtein, 1966) for strings, as both methods are based on cost matrices and dynamic programming. Similarly to Bellet et al. (2011), our theoretical results are derived using uniform stability. However, two significant differences distinguish the approaches. First, our problem is more complex: we are considering multiple features, while their case is the equivalent of univariate time series. Second, concerning the generalization bound, our case is tighter, because our bound does not depend on the length of time series or of the alignment, while they depend on the squared length of the alignment.

The rest of this chapter is organized as follows. Section 5.2 contains the main formulation of SLTS. We start by introducing the similarity function that we propose to learn for time series, before showing how to learn it while maximizing its (ϵ, γ, τ)-goodness. Section 5.3 features a theoretical analysis of SLTS based on the framework of uniform stability, which

leads to the derivation of a generalization bound. In Section 5.4, we present an experimental study proving the advantages of SLTS, as well as an extended comparison to state of the art methods. We conclude our contribution in Section 5.5, where we also present future directions of research.

5.2 Learning (ϵ, γ, τ)-Good Similarities for Time Series

This section presents a novel method for learning temporal similarity functions based on the (ϵ, γ, τ)-good framework. We start by defining the bilinear similarity for multivariate time series, which takes into account an alignment. Then, we present our convex formulation for learning it, while improving its goodness for linear classification.

We start by making the notations that will be used throughout this section. Let $\mathbf{A} \in \mathbb{R}^{t_\mathbf{A} \times d}$ be a multivariate time series of length $t_\mathbf{A}$ and dimension d. We denote by $\mathcal{X}$ the space of all time series with d features, but of different (finite) lengths. Now consider the following binary classification problem: we are given labeled multivariate time series $(\mathbf{A}, y)$ drawn from a distribution P over $\mathcal{X} \times \{+1, -1\}$, possibly of different lengths, but of same dimension d. Without loss of generality, we consider each time series to be normalized as $\|\mathbf{a}_i\|_2 = 1, i \in \{1 \dots t_\mathbf{A}\}$, for all $\mathbf{A} \in \mathcal{X}$. The results that we develop in the rest of this chapter hold for bounded multivariate time series in general, up to a normalization factor.

5.2.1 Bilinear Similarity for Time Series

We now introduce the bilinear similarity function that will be used in SLTS. For a pair of time series $\mathbf{A}$ and $\mathbf{B}$ both from $\mathcal{X}$, let $\mathbf{C}_\mathbf{M}(\mathbf{A}, \mathbf{B}) \in \mathbb{R}^{t_\mathbf{A} \times t_\mathbf{B}}$ be a pairwise matrix of the costs of aligning a time moment in $\mathbf{A}$ to one in $\mathbf{B}$ under the metric parameterized by the matrix $\mathbf{M}$. As we use a similarity function, $\mathbf{C}_\mathbf{M}(\mathbf{A}, \mathbf{B})$ represents the affinity scores that we want to maximize: the higher the score, the more the two time moments are similar. We refer to the rows of $\mathbf{A}$ as $\mathbf{a}_1, \dots, \mathbf{a}_{t_\mathbf{A}}$ and those of $\mathbf{B}$ as $\mathbf{b}_1, \dots, \mathbf{b}_{t_\mathbf{B}}$. We will focus on a bilinear similarity, where the affinity between time moment i from series $\mathbf{A}$ and time moment j from series $\mathbf{B}$ is expressed as:

$$\mathbf{C}_\mathbf{M}(\mathbf{A}, \mathbf{B})_{i,j} = \mathbf{a}_i^T \cdot \mathbf{M} \cdot \mathbf{b}_j, \tag{5.1}$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is the matrix parameterizing the metric. For the pair of indices i and j, the affinity is equivalent to computing the generalized cosine similarity, as $\mathbf{a}_i$ and $\mathbf{b}_j$ are already normalized. The overall affinity matrix can be written as:

$$\mathbf{C}_\mathbf{M}(\mathbf{A}, \mathbf{B}) = \mathbf{A} \cdot \mathbf{M} \cdot \mathbf{B}^T.$$

$\mathbf{C}_\mathbf{M}$ can be used to compute the optimal alignment between two time series using DTW. Given this affinity matrix, let $\pi_\mathbf{AB}$ be an alignment between $\mathbf{A}$ and $\mathbf{B}$ of length $t_\mathbf{AB}$. Recall that, following our definition of alignment between time series, $\pi_\mathbf{AB}$ contains the pairs of indices from $\mathbf{A}$ and $\mathbf{B}$ respectively representing the time moments that are aligned. Now

let us rewrite the alignment as a binary matrix $\mathbf{Y} \in \{0,1\}^{t_\mathbf{A} \times t_\mathbf{B}}$ encoding it: $\mathbf{Y}^{ij}_{\mathbf{A},\mathbf{B}} = 1$ if the pair of indices (i, j) is part of $\pi_{\mathbf{AB}}$, and zero otherwise. The matrix $\mathbf{Y}_{\mathbf{A},\mathbf{B}}$ thus contains $t_{\mathbf{AB}}$ non-zero values. Computing the score of aligning $\mathbf{A}$ and $\mathbf{B}$ from the affinity matrix and the alignment can be written as the following similarity function:

$$\begin{aligned} \mathrm{K}_\mathbf{M}(\mathbf{A}, \mathbf{B}) &= \mathrm{tr}(\mathbf{C}_\mathbf{M}(\mathbf{A}, \mathbf{B})^T \cdot \mathbf{Y}_{\mathbf{AB}})/t_{\mathbf{AB}} \\ &= \mathrm{tr}(\mathbf{B} \cdot \mathbf{M}^T \cdot \mathbf{A}^T \cdot \mathbf{Y}_{\mathbf{AB}})/t_{\mathbf{AB}}. \end{aligned}$$

The previous equation sums the costs of from $\mathbf{C}_\mathbf{M}(\mathbf{A}, \mathbf{B})$, but only for the pairs of points that are aligned by $\mathbf{Y}_{\mathbf{AB}}$. When computing the product between the affinity matrix and the alignment, the scores of the pairs of points that are aligned end up on the main diagonal of the resulting matrix. Applying the trace operator sums only these diagonal values, while discarding the others. As the value of the similarity is cumulative, we normalize it w.r.t. the length of the alignment in order to remove the bias created by very long alignments.

To compute $\mathrm{K}_\mathbf{M}$, we will not consider the optimal alignment with respect to $\mathbf{M}$, which would make it sensitive to changes in the similarity parameters. Instead, we will compute it through DTW using the affinity matrix in Equation (5.1) and replacing $\mathbf{M}$ with the identity matrix. Determining $\mathbf{C}_\mathbf{I}(\mathbf{A}, \mathbf{B})$ is thus equivalent to computing the scalar product between all combinations of time moments between $\mathbf{A}$ and $\mathbf{B}$.

Using $\mathrm{K}_\mathbf{M}$ as similarity function to compare multivariate time series allows us to take advantage of the ideal alignment, while considering an improved weighting of the features and cross-features for each time moment. Note that a bilinear form has the capacity to compare time series that have a different number of features, provided that the shape of the matrix $\mathbf{M}$ is adapted. An important property is that the metric matrix $\mathbf{M}$ does not have to be PSD nor symmetric. Moreover, the similarity function $\mathrm{K}_\mathbf{M}$ is linear in $\mathbf{M}$, meaning that it can be optimized directly. We shall now discuss a method for learning $\mathbf{M}$ from data.

5.2.2 Learning Good Similarities

Our objective is to learn the matrix $\mathbf{M}$ that parameterizes the similarity function $\mathrm{K}_\mathbf{M}$ for usage in classification. To this end, a labeled training set $\mathcal{S}$ of m time series $\{(\mathbf{A}_i, y_i)\}_{i=1}^m$ drawn accordingly to P is available, together with a set $\mathcal{L}$ of n landmarks $\{(\mathbf{B}_j, y'_j)\}_{j=1}^n$ coming from the same distribution. When optimizing the goodness criterion, we do so with respect to the set of landmarks $\mathcal{L}$, which we suppose fixed. Two heuristics for choosing them from data are discussed in the experiments (Section 5.4). As opposed to the previous chapter, the landmarks are this time labeled, and their class information will be used to create positive and negative pairs over the whole training set $\mathcal{S}$.

We want to optimize the (ϵ, γ, τ)-goodness of the proposed similarity function as presented

in Definition 4.3. Rewriting its first condition for our problem yields:

$$\mathbb{E}_{(\mathbf{A},y)}\left[\left[1-\mathbb{E}_{(\mathbf{B},y'),R(\mathbf{B})}\left[yy'\,\mathrm{K}_{\mathbf{M}}(\mathbf{A},\mathbf{B}))|R(\mathbf{B})\right]/\gamma\right]_+\right]\leq\epsilon.$$

Unfortunately, we do not have access to the expected values. We propose to estimate its empirical value and improve it instead in a regularized risk minimization setting. First, the empirical loss suffered by one example $(\mathbf{A}, y)$ is:

$$\ell(\mathbf{M},(\mathbf{A},y),\mathcal{L})=\left[1-\frac{1}{n\gamma}\sum_{(\mathbf{B},y')\in\mathcal{L}}yy'\,\mathrm{K}_{\mathbf{M}}(\mathbf{A},\mathbf{B})\right]_+$$

Learning the similarity that satisfies Definition 4.3 is equivalent to learning the entries of the matrix $\mathbf{M}$ that parameterizes it and is done by solving the following optimization problem over $\mathbf{M}$:

$$\min_{\mathbf{M}}\frac{1}{m}\sum_{(\mathbf{A},y)\in\mathcal{S}}\left[1-\frac{1}{n\gamma}\sum_{(\mathbf{B},y')\in\mathcal{L}}yy'\,\mathrm{K}_{\mathbf{M}}(\mathbf{A},\mathbf{B})\right]_+ +\lambda\left\|\mathbf{M}\right\|_{\mathcal{F}}^2, \tag{5.2}$$

where λ is the regularization parameter. Tuning it controls the trade-off between fitting the data and limiting the complexity of the hypothesis. In order to avoid overfitting, the objective function is regularized with the squared Frobenius norm of the matrix $\mathbf{M}$. This norm allows to control the complexity of $\mathbf{M}$ in a loose way, without strongly pushing its entries to zero. Using this regularizer will allow us to provide theoretical guarantees for the proposed approach through uniform stability. Notice that the similarity function $\mathrm{K}_{\mathbf{M}}$ is linear in $\mathbf{M}$, making the whole formulation convex. Problem (5.2) is a quadratic program (QP) and can easily be solved. We call the proposed method Similarity Learning for Time Series (SLTS). After solving Problem (5.2), $\mathrm{K}_{\mathbf{M}}$ is plugged in Equation (4.1) in order to learn the linear separator $\boldsymbol{\alpha}$.

SLTS is different from the standard metric learning setting in that it only computes similarities with respect to chosen landmarks, not the entire training set. Moreover, the constraints generated this way over pairs of points do not have to be entirely satisfied, but only on average for each training point. The total number of constraints generated is linear in the size of the dataset, not quadratic, as it is the case with pairs. Having a formulation based on landmarks also implies that the optimal alignment based on DTW only needs to be computed for the data points with respect to the set of landmarks. As computing DTW is expensive (recall its quadratic complexity in the lengths of the time series), the lower the number of landmarks, the faster the computation. This is a major advantage of our approach when compared to classic methods like k-NN and SVM, which have to compute the entire empirical similarity map for all the pairs of points in the dataset. When fixing the number of landmarks to a value that is much smaller than the size of the dataset and independent of it, the complexity of computing all the similarity values for SLTS is linear in the size of the dataset, while for k-NN and SVM it is quadratic.

5.3 Theoretical Guarantees

In this section, we present a theoretical analysis of our similarity learning approach based on the notion of uniform stability (Bousquet & Elisseeff, 2002). First, we prove that SLTS is stable, property that we then use to derive a consistency bound for the metric. The main result in this section is presented in Theorem 5.7. We conclude the analysis by a discussion on the main properties of the bound.

Learning the metric by solving Problem (5.2) places our approach in the framework of (ϵ, γ, τ)-good similarity functions, which enforces the theoretical guarantees from Theorem 4.4, that is a bounded error for the learned linear classifier with respect to the properties of $\mathrm{K}_{\mathbf{M}}$. Additionally, the bound that we derive in this section on the consistency of $\mathbf{M}$ provides a link between the empirical loss we are minimizing under regularization in Equation (5.2) and the value we want to minimize, the true loss.

We make the following notations. According to Problem (5.2), SLTS minimizes the empirical risk of the learned matrix $\mathbf{M}$ over the whole training set $\mathcal{S}$:

$$R^{\ell}_{\mathcal{S}}(\mathbf{M}) = \frac{1}{m} \sum_{(\mathbf{A},y)\in\mathcal{S}} \ell(\mathbf{M}, (\mathbf{A}, y), \mathcal{L}).$$

Following Definition 4.3, the error that the algorithm should minimize is the true risk:

$$R^{\ell}_{P}(\mathbf{M}) = \mathbb{E}_{(\mathbf{A},y)\sim P}\left[\ell(\mathbf{M}, (\mathbf{A}, y), \mathcal{L})\right].$$

As the set of landmarks is fixed, we will omit it from the notation of the loss function to simplify the notation. SLTS does not use the classic setting for metric learning, based on pairs or triplets, but the fixed set of landmarks $\mathcal{L}$. We are thus able to derive our theoretical results based on the standard setting for uniform stability, without its adaptation to pairs of points specific to metric learning (Jin et al., 2009). Recall the intuition behind uniform stability: an algorithm is stable if its outcome is not strongly influenced by small changes in the input. In this framework, the notion of marginal variation is conveyed by the replacement of one example by a new one from the same distribution P. We will denote by $\mathcal{S}^i$ the training set obtained from $\mathcal{S}$ by changing the ith example. Similarly, let $\mathbf{M}^i$ be the solution to SLTS when solved for training set $\mathcal{S}^i$. To prove the stability of SLTS, we first need to show that the considered loss function is bounded and l-lipschitz: the smaller l, the more stable the algorithm. We do so in Lemmas 5.1 and 5.2. All the proofs of the Lemmas in this section are presented in Appendix B.2.

Lemma 5.1 (Bound on the loss function)**.** *Let* $(\mathbf{A}, y)$ *be an example and* $\mathbf{M}_{\mathcal{S}}$ *the solution to Problem* (5.2)*. Then*

$$\ell(\mathbf{M}_{\mathcal{S}}, (\mathbf{A}, y)) \leq \frac{\sqrt{2d}}{\gamma\sqrt{\lambda}}.$$

In Chapter 4, we have used l-lipschitzness to bound the value of a similarity function w.r.t. the difference between two examples, in order to prove algorithmic robustness. For uniform stability, we prove the l-lispchitzness of the loss function w.r.t. the variation in $\mathbf{M}$ in the

next lemma.

Lemma 5.2 (l-lipschitz continuity)**.** *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be two matrices and* $(\mathbf{A}, y)$ *an example. The loss function* ℓ *is* l*-lipschitz w.r.t.* $\mathbf{M}$ *with* $l = \frac{\sqrt{2d}}{\gamma}$*, that is:*

$$|\ell(\mathbf{M}, (\mathbf{A}, y)) - \ell(\mathbf{M}', (\mathbf{A}, y))| \leq l \left\|\mathbf{M} - \mathbf{M}'\right\|_{\mathcal{F}}.$$

The property of l-lipschitzness implies that the loss variation is proportional to the difference between $\mathbf{M}$ and $\mathbf{M}'$. The lemma we present next is used for the proof of the uniform stability of an algorithm. Consider the following notation for the objective function of SLTS (Equation (4.3)):

$$F_{\mathcal{S}}(\mathbf{M}) := R_{\mathcal{S}}(\mathbf{M}) + \lambda \left\|\mathbf{M}\right\|_{\mathcal{F}}^2.$$

Lemma 5.3. *Let* $F_{\mathcal{S}}(\cdot)$ *and* $F_{\mathcal{S}^i}(\cdot)$ *be the functions to optimize,* $\mathbf{M}$ *and* $\mathbf{M}^i$ *their corresponding minimizers, and* λ *the regularization parameter used. Let* $\Delta\mathbf{M} = \mathbf{M} - \mathbf{M}^i$*. Then we have, for* $t \in [0, 1]$*:*

$$\left\|\mathbf{M}\right\|_{\mathcal{F}}^2 - \left\|\mathbf{M} - t\Delta\mathbf{M}\right\|_{\mathcal{F}}^2 + \left\|\mathbf{M}^i\right\|_{\mathcal{F}}^2 - \left\|\mathbf{M}^i + t\Delta\mathbf{M}\right\|_{\mathcal{F}}^2 \leq \frac{2lt}{\lambda m} \left\|\Delta\mathbf{M}\right\|_{\mathcal{F}}.$$

We can now prove that our approach has uniform stability.

Theorem 5.4 (Stability of SLTS)**.** *Given a training sample* $\mathcal{S}$ *of* m *examples drawn i.i.d. from* P*, our algorithm SLTS has uniform stability in* κ/m *with* $\kappa = \frac{4d}{\gamma^2\lambda}$*.*

Proof. By setting $t = \frac{1}{2}$ in Lemma 5.3, we obtain after some computations:

$$\frac{1}{2} \left\|\Delta\mathbf{M}\right\|_{\mathcal{F}}^2 \leq \frac{k}{\lambda m} \left\|\Delta\mathbf{M}\right\|_{\mathcal{F}},$$

which implies:

$$\left\|\Delta\mathbf{M}\right\|_{\mathcal{F}} \leq \frac{2k}{\lambda m}.$$

Since our loss is l-lipschitz, we have:

$$|\ell(\mathbf{M}, (\mathbf{A}, y)) - \ell(\mathbf{M}^i, (\mathbf{A}, y))| \leq l \left\|\Delta\mathbf{M}\right\|_{\mathcal{F}} = \frac{2l^2}{\lambda m}$$

For this loss function, $l = \frac{\sqrt{2d}}{\gamma}$, and setting $\kappa = \frac{4d}{\gamma^2\lambda}$ proves the lemma. □

Having now shown the uniform stability of SLTS, we are ready to derive the generalization bound. For this, Lemmas 5.5 and 5.6 are necessary, providing bounds on quantities that intervene in the proof of the bound. Let $\mathcal{E}_{\mathcal{S}} = R_P^{\ell}(\mathbf{M}) - R_{\mathcal{S}}^{\ell}(\mathbf{M})$. We need to bound the quantities $\mathbb{E}_{\mathcal{S}}[\mathcal{E}_{\mathcal{S}}]$ and $|\mathcal{E}_{\mathcal{S}} - \mathcal{E}_{\mathcal{S}^i}|$.

Lemma 5.5. *For a learning method of estimation error* $\mathcal{E}_{\mathcal{S}}$ *and satisfying a uniform stability of* κ/m*, we have:*

$$\mathbb{E}_{\mathcal{S}}[\mathcal{E}_{\mathcal{S}}] \leq \frac{\kappa}{m}.$$

Lemma 5.6. *For any metric* $\mathbf{M}$ *learned by solving Problem* (5.2) *on a training set* $\mathcal{S}$ *of* m *samples, and a loss function* ℓ *bounded according to Lemma 5.1, we have:*

$$|\mathcal{E}_{\mathcal{S}} - \mathcal{E}_{\mathcal{S}^i}| \leq \frac{2\kappa}{m} + \frac{\sqrt{2d}}{m\gamma\sqrt{\lambda}}.$$

We can now present our main theoretical result, the consistency bound for the metric learned by SLTS:

Theorem 5.7 (Generalization bound using uniform stability)**.** *With probability* $1-\delta$*, for any matrix* $\mathbf{M}_{\mathcal{S}}$ *learned by solving SLTS, we have:*

$$R_P^{\ell}(\mathbf{M}_{\mathcal{S}}) - R_{\mathcal{S}}^{\ell}(\mathbf{M}_{\mathcal{S}}) \leq \frac{4d}{\gamma^2\lambda m} + \left(\frac{4d}{\gamma^2\lambda} + \frac{1}{\gamma}\sqrt{\frac{2d}{\lambda}}\right)\sqrt{\frac{2\log\frac{2}{\delta}}{m}}.$$

Proof. Using McDiarmid's inequality and Lemma 5.6, we can write:

$$\Pr[\mathcal{E}_{\mathcal{S}} - \mathbb{E}[\mathcal{E}_{\mathcal{S}}] \geq \epsilon] \leq 2\exp\left(-\frac{2\epsilon^2}{m\left(\frac{2\kappa+p}{m}\right)^2}\right). \tag{5.3}$$

By setting $\delta = 2\exp\left(-\frac{2\epsilon^2}{m\left(\frac{2\kappa+p}{m}\right)^2}\right)$ in Inequality (5.3), we obtain:

$$\epsilon = \sqrt{\frac{2}{m}\left(\frac{4d}{\gamma^2\lambda} + \frac{1}{\gamma}\sqrt{\frac{2d}{\lambda}}\right)^2\log\frac{2}{\delta}}.$$

Then, with probability $1-\delta$

$$\begin{aligned}\mathcal{E}_{\mathcal{S}} = R_P^{\ell}(\mathbf{M}_{\mathcal{S}}) - R_{\mathcal{S}}^{\ell}(\mathbf{M}_{\mathcal{S}}) &< \mathbb{E}[\mathcal{E}_{\mathcal{S}}] + \epsilon \\ R_P^{\ell}(\mathbf{M}_{\mathcal{S}}) &< R_{\mathcal{S}}^{\ell}(\mathbf{M}_{\mathcal{S}}) + \frac{\kappa}{m} + \epsilon.\end{aligned}$$

Replacing the values of κ and ϵ in the previous inequality yields the bound. □

The result from Theorem 5.7 shows the consistency of the proposed similarity learning approach. The bound converges with a standard rate of $\mathcal{O}(1/\sqrt{m})$ in the number of samples. The tightness of the bound is inversely proportional to the size of the margin γ, supported also by the intuition that a better separation of the classes is more robust to new examples arriving, keeping the empirical error closer to the true error. An important feature of this bound is its independence from the lengths of the time series and of the alignments considered. As in practice these values can be elevated, introducing even a linear dependence on them would deteriorate the quality of the bound, making it loose and uninformative. Overall, the bound strongly resembles PAC bounds derived using the uniform stability for methods dealing with feature vectors, even though multivariate time series have an additional dimension. This means that SLTS does not suffer any penalty in its generalization capacity from working with multivariate time series instead of feature

Table 5.1: Properties of the datasets used in the experimental study.

Dataset	#Instances	Length	#Features	#Classes
Japanese vowels	640	7-29	12	9
Auslan	675	47-95	22	25
Arabic digits	8800	4-93	13	10
Robot exec. failure				
LP1	88	15	6	4
LP2	47	15	6	5
LP3	47	15	6	4
LP4	117	15	6	3
LP5	164	15	6	5

vectors. According to (Verma & Branson, 2015), the presence of the number of features d in the numerator of the bound is to be expected and shows that the approach may suffer from the curse of dimensionality. High values of d can be compensated by increasing either the size of $\mathcal{S}$, or the value of the regularization parameter λ, present in the denominator. SLTS minimizes the empirical error of the (ϵ, γ, τ)-good framework, thus reducing the error rate ϵ. By plugging the metric learned by SLTS into the framework, we obtain a guarantee on the performance of the associated linear classifier.

5.4 Experimental Validation

In this section, we present the results of the experiments conducted to evaluate the performance of the proposed method. In the first experiment, we show that learning the matrix $\mathbf{M}$ brings additional information w.r.t. a standard metric for linear classification. We also analyze the influence of the number of landmarks on SLTS. The second study provides a comparison of SLTS to the state of the art algorithms while the third part illustrates the capacity of SLTS to learn a discriminant metric in the feature space created by the landmarks. The last experiment analyzes different heuristics for selecting landmarks and their impact on performance. We conduct the experimental study on multivariate time series datasets coming from UCI Machine Learning Repository (Lichman, 2013), containing between 47-8800 instances. The characteristics of the datasets are presented in Table 5.1.

5.4.1 Experimental Setting

Compared methods We evaluate the following classic algorithms:

- Standard nearest neighbor classifier (1NN);
- Linear SVM under L_2 regularization;
- Linear classifier from Balcan et al. (2008b), presented in Equation (4.2) (called BBS from now on);
- LDMLT (Mei et al., 2015) with a nearest neighbor classifier;

Table 5.2: Classification accuracy (%) with confidence interval at 95%.

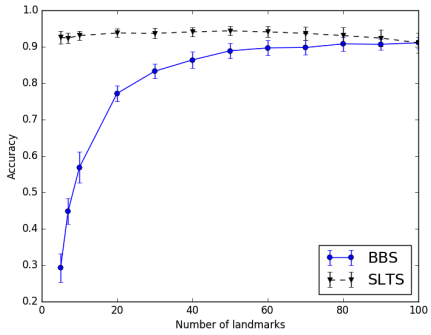
Method	Japanese vowels	Auslan	Arabic digits	Robot exec. failure	Avg.
1NN	93.8	77.8±2.1	94.7	68.8±7.5	92.1
LDMLT	97.3	**95.0**±1.3	96.9	**71.9**±7.0	95.6
SVM	**97.8**±0.1	92.6±0.1	93.3±0.0	60.6±6.5	92.2
BBS	97.1±0.5	91.1±1.6	96.4±0.3	66.9±10.6	94.7
SLTS	97.1±0.4	91.1±2.7	**97.9**±0.4	67.0±7.8	**95.8**

- SLTS, the similarity learning method proposed in this chapter, which is then used to learn a global linear classifier using the formulation in Balcan et al. (2008b).

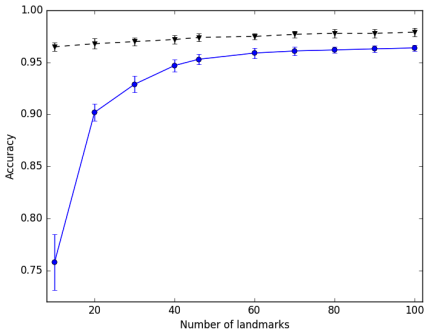
Data processing and parameter settings To propose a fair comparative study, all the methods that do not learn a metric use the proposed bilinear form as similarity function (with $\mathbf{M}$ set to the identity matrix) computed with the DTW alignment on the scalar product, that is $\mathrm{K}_{\mathbf{I}}$ based on $\mathbf{C}_{\mathbf{I}}$. As confirmed in the following experiments, landmarks are randomly chosen for BBS and SLTS. We use all the classifiers in their binary version, in a one-vs-all setting. We recall here that each time moment is normalized to ensure the L_2 norm equals 1. For this experimental study, we have access to a standard training/test partitioning for Japanese vowels and Arabic digits datasets, while Robot execution failure (LP1-5) and Auslan are randomly split to 70% training/30% test data. For all datasets, we retain 30% of the training set for hyperparameter tuning. We perform experiments on 10 different splits and present the average result with a 95% confidence interval. Cross-validation is performed to tune the following parameters: $C \in \{2^{-6}, \ldots, 2^{9}\}$ for SVM, $\gamma \in \{10^{-4}, \ldots, 10^{1}\}$ for BBS, both when used separately or joint to SLTS, and $\lambda \in \{0.1, 1, 10\}$ for SLTS.

5.4.2 Experimental Results

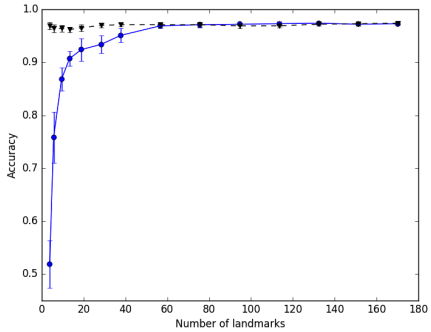
Behavior of SLTS and impact of the number of landmarks As stated before, BBS and SLTS use the same formulation to learn the linear classifier. In this first experiment, we (i) show that SLTS improves classification performance compared to BBS and (ii) analyze the influence of the quantity of landmarks on the accuracy obtained for BBS and SLTS. We consider the range of up to 50% of the size of the training set as landmarks for small datasets, or up to 100 landmarks for the others. The results of this study are presented in Figure 5.1. The accuracy of SLTS is almost always higher than that of BBS independently of the number of landmarks, showing the improvement that can be obtained through similarity learning. When a reasonable quantity of data is available (Figures 5.1(a)-5.1(c)), SLTS achieves a performance close to its best value even with a few landmarks. Our method thus performs well even with a low quantity of data, considerably reducing the computation time for learning the similarity. Overall, BBS has difficulties providing a good classifier based on a small number of landmarks, but the results of the method significantly improve with more landmarks. We explain the high variability of the results of BBS and SLTS on LP1-LP5 by the small sizes of the tasks.



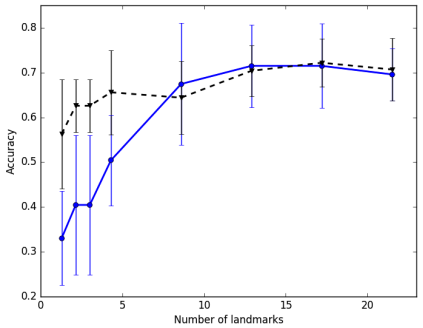
(a) Auslan



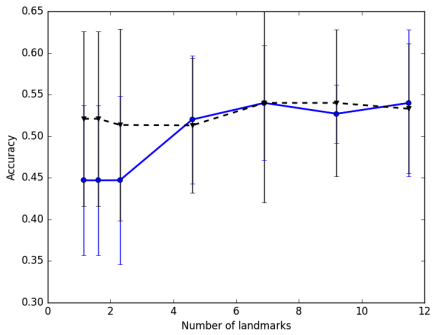
(b) Arabic digits



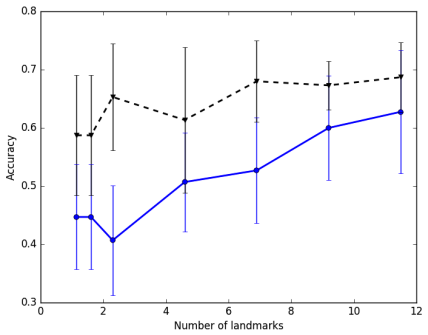
(c) Japanese vowels



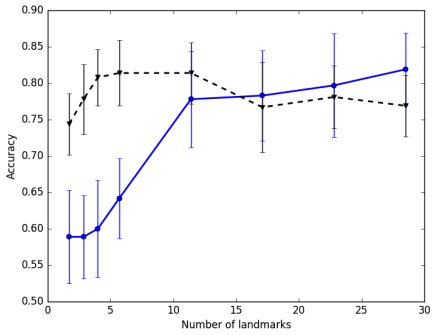
(d) LP1



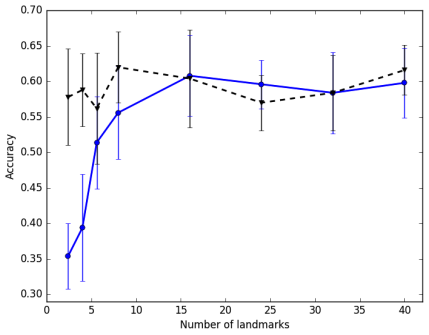
(e) LP2



(f) LP3



(g) LP4



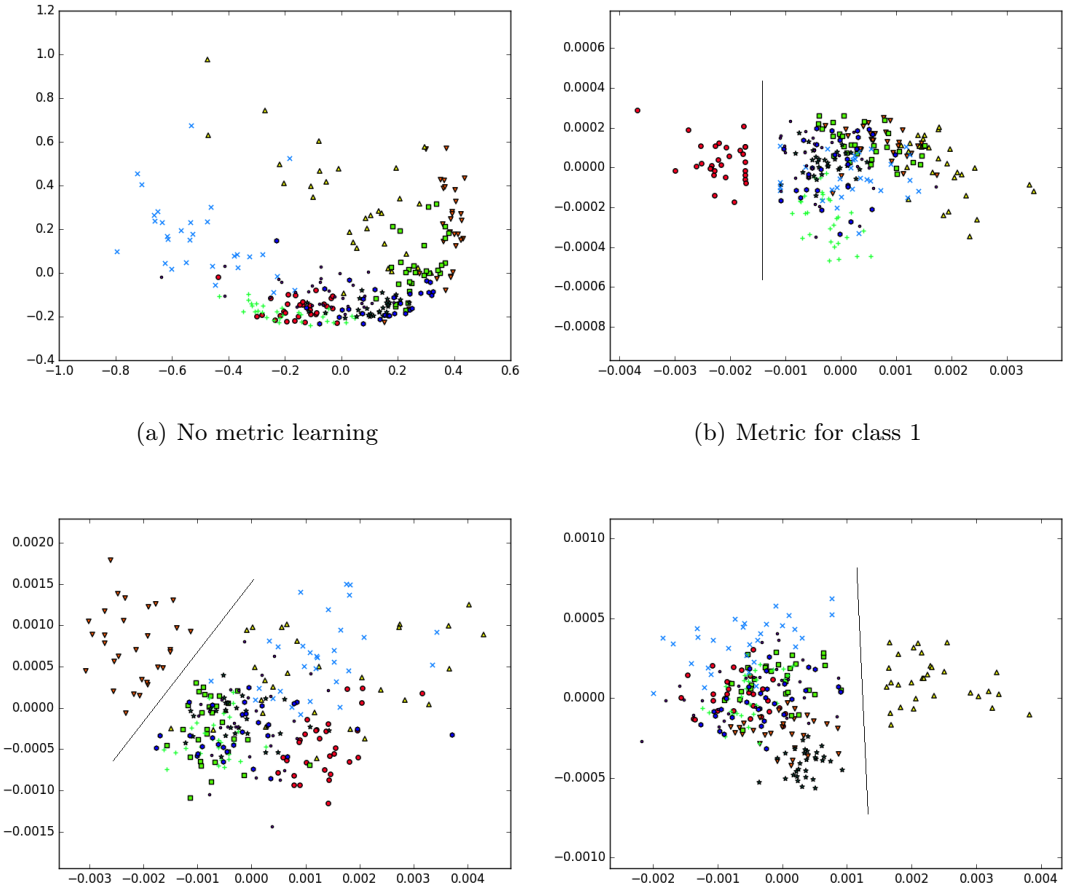
(h) LP5

Figure 5.1: Classification accuracy of BBS and SLTS w.r.t. the number of landmarks.

Classification performance comparison The results of the comparison of SLTS and BBS with other methods are displayed in Table 5.2. For this second experiment, both SLTS and BBS are based on the maximum number of landmarks from the previous experiment. Note that confidence interval in the table values means that the train/test split of the data is already provided, and the output of the method is deterministic. As one can observe, among global methods relying on a linear classification (i.e., SLTS, BBS, and SVM), both SLTS and BBS perform better than SVM (they are on a par on Japanese vowels, slightly below on Auslan, and above on Arabic digits and Robot execution failure). Using a Student t-test for paired samples on the average reveals that SLTS is significantly better than BBS and SVM. This shows the usefulness of the (ϵ, γ, τ)-good framework as well as the importance of metric learning in this framework. The comparison of SLTS with local methods (as 1NN and LDMLT) yields more contrasted results. On all datasets except Robot exec. failure, 1NN is significantly below SLTS according to a Student t-test. However, compared to LDMLT, SLTS is on a par on Japanese vowels, below on Auslan and Robot execution failure, and above on Arabic digits (a Student t-test on the average does not reveal any significant difference between the two methods). LDMLT relies on both a local method and a metric learned, which suggests again that learning a metric is beneficial on these datasets. This said, LDMLT learns a distance, whereas all the other methods rely on a similarity. The comparison between the two should thus be taken with caution as distances and similarities can yield very different results (Qamar & Gaussier, 2009a).

Visualization of the similarity space To illustrate the transformation induced in the feature space by learning the metric, we propose a visualization experiment on the Japanese vowels dataset using 10 landmarks chosen randomly. We compute the value of the similarity function $\mathrm{K}_{\mathbf{M}}$ for all the data w.r.t. the landmarks, first without metric learning ($\mathbf{M} = \mathbf{I}$), then with the metric learned for each of the 9 classes. In all the cases, we apply PCA (Jolliffe, 1986) to the values of the similarity function and plot the first two components. We thus obtain a 2D representation of the feature space, of which we present in Figure 5.2 the case of the initial feature space and that of the metric learned for the first three classes. In similarity space with no metric learning (Figure 5.2(a)), all the data points are mixed, independently of their label. In Figures 5.2(b)-5.2(d), each metric linearly separates the class it has learned to discriminate from the others. For the learned similarities, the first two components of PCA explain around 98% of the variance, while with no similarity learning this value is around 86%. This study proves that learning an (ϵ, γ, τ)-good similarity function changes the representation space towards better class discrimination, making it suitable for learning a large margin linear separator.

Heuristics for choosing the landmarks We have previously assumed we have access to a set of landmarks for the construction of the feature space. We will now discuss two heuristics for choosing the most representative points in the training set as landmarks, before presenting experimental results concerning the performance of each of these methods.
K-Medoids (Kaufman & Rousseeuw, 1987) is a classical clustering technique. The resulting medoids representing the clusters are points of the initial dataset, that will be



(a) No metric learning

(b) Metric for class 1

(c) Metric for class 2

(d) Metric for class 3

Figure 5.2: PCA (first two components) in the similarity space for Japanese vowels. Each class becomes linearly separable from the others when using its corresponding metric.

subsequently used as landmarks. **Dselect** (Kar & Jain, 2011) was proposed as a landmarks selection algorithm which optimizes a criterion of diversity. Starting with a randomly chosen landmark, at each iteration the algorithm greedily adds to the set of landmarks the training point that is least similar to the ones already selected (see Algorithm 5.1). Note that for both selection heuristics the number of landmarks needs to be set in advance. Also, none of these methods exploits the information from the labels of the time series. In the case where no prior information is available for the classification task, the set of landmarks can also be selected randomly from the training set, with the risk of relying upon non informative landmarks.

We now present in Tables 5.3 and 5.4 the classification results after learning the similarity with SLTS on landmarks selected using the presented heuristics. DSelect and KMedoids are compared against landmarks selected randomly as baseline, in order to determine if they are indeed informative. We perform these experiments on two small datasets, Japanese vowels and LP1. The mass of chosen landmarks is selected as a percentage of the total size of the training set and goes up to 50%. For Japanese vowels (Table 5.3), all three methods perform almost the same for all amounts of landmarks. DSelect reaches its best

Algorithm 5.1 Dselect heuristic

Input A training set $\mathcal{S}$, number of landmarks n
Output A set $\mathcal{L}$ of n landmarks
  $\mathcal{L} \leftarrow$ random element of $\mathcal{S}$
  for $i = 2$ to n **do**
    $z \leftarrow \arg\min_{z=(\mathbf{x},y)\in\mathcal{S}} \sum_{z'=(\mathbf{x}',y')\in\mathcal{L}} \mathrm{K}(\mathbf{x},\mathbf{x}')$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{z\}$
    $\mathcal{S} \leftarrow \mathcal{S} \setminus \{z\}$
  end for
  return $\mathcal{L}$

Table 5.3: Classification accuracy for landmarks selection methods on Japanese vowels.

Lmks	DSelect	KMedoids	Random
3%	**96.8**±0.6	96.3±0.6	96.4±0.7
5%	**96.5**±0.6	95.9±0.7	96.4±0.6
7%	**97.0**±0.2	96.3±0.4	96.2±0.5
10%	**97.3**±0.3	96.2±0.5	96.5±0.6
15%	**97.1**±0.2	96.4±0.4	97.0±0.4
20%	**97.1**±0.3	96.8±0.4	**97.1**±0.4
30%	96.7±0.4	97.0±0.3	**97.1**±0.4
40%	97.0±0.4	96.9±0.3	**97.1**±0.4
50%	96.8±0.3	**96.9**±0.3	**96.9**±0.4

performance when the selected points represent 10-20% of the training set, while KMedoids works best around 30% mass of landmarks. Overall, DSelect and Random heuristics yield better performance than KMedoids. The results using Random show that SLTS can learn well ever when no computational effort is put into choosing the landmarks. In the case of LP1 (Table 5.4), and in contrast to the Japanese vowels dataset, the performance of all the heuristics improves when increasing the number of landmarks. The best results are obtained for 40% mass of landmarks in the case of DSelect and Random, and 50% for KMedoids. For this dataset, the results are less stable, inducing larger confidence intervals. For this reason, even though the best accuracy is attained by DSelect, its improvement over Random is not necessarily significant. KMedoids is this time also the least performant

Table 5.4: Classification accuracy for landmarks selection methods on LP1.

Lmks	DSelect	KMedoids	Random
3%	48.5±6.2	50.7±9.4	**56.3**±12.2
5%	**67.8**±9.7	63.3±10.1	62.6±5.9
7%	**67.8**±9.7	63.3±10.1	62.6±5.9
10%	**68.5**±9.6	65.2±5.5	65.6±9.4
15%	67.0±6.4	**69.3**±6.8	68.5±10.0
20%	**71.1**±5.6	68.9±5.8	64.4±8.1
30%	**70.4**±7.2	66.7±6.8	**70.4**±5.7
40%	**74.4**±7.9	70.4±6.5	72.2±5.3
50%	**73.0**±7.0	71.5±8.5	70.7±7.0

heuristic.

KMedoids and DSelect have by themselves a computational complexity that is not to be ignored when working on large datasets. Even so, their main disadvantage for time series is not the algorithmic complexity in itself, but the necessary precomputations. One needs to compute the value of the similarity function for all pairs of time series, including the alignment, in order to be able to apply these heuristics. This limitation goes directly against the main advantage of working with methods based on landmarks, like SLTS. In view of this aspect and the previous experimental results, we have only considered the Random heuristic when comparing SLTS against state of the art algorithms on bigger datasets.

5.5 Conclusion

The last contribution of this thesis addresses the problem of improving the performance when learning a global linear classifier for multivariate time series. In the present chapter, this problem was tackled by learning a bilinear similarity under the optimal alignment provided by DTW. SLTS is an efficient convex formulation which improves the (ϵ, γ, τ)-goodness of the similarity with respect to a set of landmarks. We thus obtain a global metric which integrates constraints from positive and negative pairs. We prove the uniform stability of our method, which allows us to derive a generalization bound for the similarity. This bound has a standard convergence rate when the size of the sample increases and the additional important advantage of being independent from the length of the time series, as well as the length of the alignment. SLTS is, to the best of our knowledge, the first approach to provide a consistency bound for the learned metric in the case of time series. The experimental study proves the usefulness of the (ϵ, γ, τ)-good framework, as well as the importance of metric learning in this setting. We also presented experiments covering different heuristics for landmarks selection, proving that random selection provides a computationally cheap alternative without strongly degrading performance.

In SLTS, the parameters of the bilinear similarity are learned under the alignment computed by DTW using a standard similarity, with no parameters. An intuitive idea is to reconsider the optimal alignment and recompute it under the new metric. We have performed this step in hope of obtaining a new, better alignment that would improve classification performance. However, in practice, recomputing the alignment using the learned metric does not necessarily improve the results, even when multiple iterations of the two steps are performed (applying DTW and learning $\mathbf{M}$). The intuition behind this behavior is that the learned matrix $\mathbf{M}$ is the solution to a convex problem which is parameterized by the initial optimal alignment. By changing the alignment, (i) $\mathbf{M}$ is no longer the solution to the problem and (ii) the new alignment is computed through a dynamic programming technique, not optimization, thus with no guarantees of minimizing the loss. A similar phenomenon was also reported in the recent work from Zhao et al. (2016).

Future work could cover an extension to other similarity functions, including some based on the Mahalanobis distance, but without enforcing the PSD constraint. Provided the

similarity function respects the necessary constraints, i.e. being bounded and l-lipschitz, the consistency bound of SLTS would be able to incorporate their properties directly. Similarly, an interesting perspective would be to study the impact regularizers with different properties on the matrix $\mathbf{M}$. For example, a sparsity-inducing norm would allow to automatically detect features that are not relevant, providing valuable information about the task at hand. This would impact SLTS with respect to the consistency bound for the metric. As mentioned previously, the framework of uniform stability can only be used with strongly convex regularizers. An equivalent PAC bound would have to be derived using an alternative framework, e.g. the Rademacher complexity.

We believe that a straightforward, but useful extension of SLTS should address learning multiple metrics. This idea is justified by the intuition that the metrics should be able to capture local temporal information: depending on the time moment, a different metric could be more appropriate. The current version of SLTS is unable to do so, as the same metric matrix is used for all time moments. Our theoretical results could be adapted to cover such a setting.

Chapter 6

Conclusion and Perspectives

In this thesis, we have given an overview of the state of development in metric learning for certain types of problems, while presenting their main limitations. We addressed a number of these limitations in our contributions, in a more general setting for feature vectors, as well as for structured data with a temporal dimension. An important focus was placed on the theoretical foundations guiding the proposals, as we believe guarantees of performance to be essential. When developing our methods, we have also tried to keep in mind the practical aspects that would make them usable.

Overall, we addressed the following limitations:

- The deficiency in theoretical results for metric learning: even though in recent years more studies have been concerned with developing theoretical analyses, two tendencies can be noticed: (i) some of these studies are only theoretical and (ii) overall, only a small number of the practical approaches in metric learning have addressed this question.
- The artificial constraints imposed on the learned metric: most methods enforce the PSD and symmetry properties, for feature vectors and time series alike, often because they learn a Mahalanobis distance. Instead, we have chosen to work with similarity functions that do not need these properties.
- The lack of results and methods in learning custom metrics for temporal data, especially for the most common type in real-world applications that is multivariate time series.

The first contribution was to propose JSL, a framework which allows to learn a similarity function at the same time as a global linear classifier. The method directly optimizes the (ϵ, γ, τ)-goodness (Balcan et al., 2008b) of the similarity function, which is directly related to its performance in classification. An important characteristic of JSL deriving from this is its semi-supervised setting. It allows the method to take advantage of unlabeled data in a different way from most semi-supervised methods, that is by using it to construct the feature space where the data will be projected. The capacity to leverage unlabeled examples makes the proposed framework particularly adapted to the real-world setting where data annotation is expensive, and only a small number of labeled instances is available. To the best of our knowledge, our method is the first one to learn the metric and the classifier at the same time partly from unlabeled data. The generality of JSL lies in the fact that it can be instantiated with a broad range of similarity functions and regularizers, thus providing

a way to solve problems with different properties. The general regularizer has the capacity to incorporate prior knowledge about the given task. We also propose an efficient method for solving the problem, which is convex under mild constraints for the similarity function. From a theoretical standpoint, we analyze JSL through the perspective of two frameworks: the algorithmic robustness and the uniform convergence with Rademacher complexity. Both analyses yield generalization bounds for the learned similarity and classifier which hold for different similarity functions and provide information about them. The extensive experimental study shows that JSL achieves better performance than a large number of other methods. The source code for JSL is publicly available under GNU/GPL 2+ license[1].

Our second contribution focuses on a particular type of data, which are multivariate time series. We proposed to learn a bilinear similarity function for a classification task. Once again, we aimed at generalizing over kernel functions by relaxing the PSD constraint, while linking the properties of the similarity to its performance in classification. This was done by learning an (ϵ, γ, τ)-good similarity based on the optimal alignment between time series computed through DTW. The problem, called SLTS, is formulated as a quadratic program, which can be solved efficiently. Working with time series is often computationally expensive, mostly because of the complexity of the data and DTW when computing the best alignment between pairs of points. We were able to reduce the number of similarity values and alignments to compute by basing the approach on landmarks (in the same sense as the (ϵ, γ, τ)-good framework) and limiting their number, while also keeping alignments fixed. We established generalization guarantees for SLTS based on uniform stability, ensuring the consistency of the metric. The obtained bound is rather tight and independent from the lengths of the time series and the alignments, resembling similar bounds for feature vectors. The experimental study shows the good properties of the metric, as well as its performance when compared to standard methods. To the best of our knowledge, SLTS is the first method performing metric learning for multivariate time series which comes with such a theoretical analysis. The source code for solving SLTS is also publicly available under GNU/GPL 2+ license[2].

There are multiple interesting perspectives for extending the methods that we have proposed. One of them is to adapt them to learning multiple metrics, each one corresponding to a different region of the space. We believe that this direction can capture additional information in the data, as well as provide an implicit way of introducing nonlinearity. Note that the (ϵ, γ, τ)-good framework includes a setting where multiple functions can be combined to obtain an overall metric that is also guaranteed to be good. Another possible extension is to adapt the proposed approaches to an online setting, where examples arrive one at a time. This would allow the methods to scale better. In practice, the first points received would be considered as landmarks for the learning process. For this setting, one could derive regret bounds with respect to the batch version.

From a theoretical standpoint, this thesis has analyzed the guarantees that can be obtained by a learned similarity function in linear classification. These results were derived based on the (ϵ, γ, τ)-good theory. Unfortunately, this framework links similarity functions to

[1]Download at http://inicolae.com/resources/jsl.zip.
[2]Download at http://inicolae.com/resources/slts.zip.

the particular case of global linear classifiers. Considering that state of the art in metric learning, be it for feature vectors or time series, is oriented towards the k-NN classifier, it would be a major result to develop theoretical analysis frameworks or particular results for certain methods concerning this local classification rule. Some results have already been established for the online and batch settings by Qamar et al. (2008) based on the voted perceptron theory of Freund & Schapire (1998).

Another promising avenue for research would be to explore settings that have not received much attention in metric learning, like unsupervised learning. This is a difficult problem, because the criterion the metric should optimize is not always evident, as is the case for performance measures as well. A first result using the uniform stability to derive theoretical guarantees for the general case of unsupervised learning algorithms was recently derived in Abou-Moustafa & Schuurmans (2015), paving the way for further research. More precisely, they focus on applications for clustering and dimensionality reduction. Unfortunately, their results depend on manually defining a criterion of evaluation for each application.

List of Publications

Articles in Peer-Reviewed International Conferences

Maria-Irina Nicolae, Éric Gaussier, Amaury Habrard, and Marc Sebban. Joint semi-supervised similarity learning for linear classification. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 594–609, 2015a.

Maria-Irina Nicolae, Marc Sebban, Amaury Habrard, Éric Gaussier, and Massih-Reza Amini. Algorithmic robustness for semi-supervised (ϵ, γ, τ)-good metric learning. In *Proceedings of the 22nd International Conference on Neural Information Processing (ICONIP)*, pages 253–263, 2015b.

Articles in Peer-Reviewed International Workshops

Maria-Irina Nicolae, Marc Sebban, Amaury Habrard, Éric Gaussier, and Massih-Reza Amini. Algorithmic robustness for learning via (ϵ, γ, τ)-good similarity functions. In *ICLR Workshop*, 2015. Poster presentation.

Articles in Peer-Reviewed National Conferences

Maria-Irina Nicolae, Éric Gaussier, Amaury Habrard, and Marc Sebban. Apprentissage joint semi-supervisé de bonnes similarités. In *French Conference on Machine Learning (CAp)*, 2015.

Maria-Irina Nicolae, Éric Gaussier, Amaury Habrard, and Marc Sebban. Apprentissage de similarités pour la classification de séries temporelles multivariées. In *French Conference on Machine Learning (CAp)*, 2016.

Appendix A

Concentration Inequalities

Concentration inequalities provide bounds on how a random variable deviates from a value, usually its expected value. The law of large numbers states that sums of independent random variables are, under very mild conditions, close to their expectation with a large probability.

McDiarmid's inequality bounds the expected value for sufficiently regular function of the variables.

Theorem A.1 (McDiarmid's inequality (McDiarmid, 1989)). *Let $X_1, \ldots, X_n$ be independent random variables taking values in the set $\mathcal{X}$. Further, let $f : \mathcal{X}^n \to \mathbb{R}$ be a function of $X_1, \ldots, X_n$ that satisfies $\forall i, \forall x_1, \ldots, x_n, x_i' \in \mathcal{X}$,*

$$|f(x_1, \ldots, x_i, \ldots, x_n) - f(x_1, \ldots, x_i' \ldots, x_n)| \leq c_i.$$

Then, for all $\epsilon > 0$,

$$\Pr[f - \mathbb{E}[f] \geq \epsilon] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}\right).$$

When the random variables X_i are bounded and f is the mean value, we obtain Hoeffding's inequality, which provides an upper bound on the probability that the sum of random variables deviates from its expected value.

Theorem A.2 (Hoeffding's inequality (Hoeffding, 1963)). *Let $X_1, \ldots, X_n$ be independent random variables respectively taking values in the intervals $[a_i, b_i]$. Further, let $\bar{X} = \frac{1}{n}\sum_{i=1}^n X_i$ Then, for all $\epsilon > 0$,*

$$\Pr[\bar{X} - \mathbb{E}[\bar{X}] \geq \epsilon] \leq \exp\left(\frac{-2\epsilon^2 n^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

Appendix B

Proofs

B.1 Proofs of Chapter 4

This appendix details proofs of Theorem 4.9, and the l-lipschitzness of the similarity functions $K^1_{\mathbf{M}}$, $K^2_{\mathbf{M}}$, and $K^3_{\mathbf{M}}$.

Proof of Theorem 4.9 (Xu & Mannor, 2010). We bound the difference between the true risk and the empirical risk in the following way:

$$
\begin{aligned}
&\left|R^{\ell}_{P}(\mathbf{M},\boldsymbol{\alpha}) - R^{\ell}_{\mathcal{S}}(\mathbf{M},\boldsymbol{\alpha})\right| = \left|\mathbb{E}_{z\sim P}\ell(\mathbf{M},\boldsymbol{\alpha},z) - \frac{1}{d_l}\sum_{i=1}^{d_l}\ell(\mathbf{M},\boldsymbol{\alpha},z_i)\right| \\
&= \left|\sum_{i=1}^{M}\mathbb{E}_{z\sim P}\left(\ell(\mathbf{M},\boldsymbol{\alpha},z)|z\in C_i\right)p(C_i) - \frac{1}{d_l}\sum_{i=1}^{d_l}\ell(\mathbf{M},\boldsymbol{\alpha},z_i)\right| \\
&\leq \left|\sum_{i=1}^{M}\mathbb{E}_{z\sim P}\left(\ell(\mathbf{M},\boldsymbol{\alpha},z)|z\in C_i\right)p(C_i) - \sum_{j=1}^{M}\mathbb{E}_{z\sim P}(\ell(\mathbf{M},\boldsymbol{\alpha},z)|z\in C_i)\frac{|N_i|}{d_l}\right| \\
&\quad + \left|\sum_{i=1}^{M}\mathbb{E}_{z\sim P}\left(\ell(\mathbf{M},\boldsymbol{\alpha},z)|z\in C_i\right)\frac{|N_i|}{d_l} - \frac{1}{d_l}\sum_{i=1}^{d_l}\ell(\mathbf{M},\boldsymbol{\alpha},z_i)\right| \qquad \text{(B.1)} \\
&= \left|\sum_{i=1}^{M}\mathbb{E}_{z\sim P}\left(\ell(\mathbf{M},\boldsymbol{\alpha},z)|z\in C_i\right)\left|p(C_i) - \frac{|N_i|}{d_l}\right|\right| \\
&\quad + \left|\frac{1}{d_l}\sum_{i=1}^{M}\sum_{z_j\in C_i}\mathbb{E}_{z\sim P}\left(\ell(\mathbf{M},\boldsymbol{\alpha},z)|z\in C_i\right) - \frac{1}{d_l}\sum_{i=1}^{M}\sum_{z_j\in C_i}\ell(\mathbf{M},\boldsymbol{\alpha},z_j)\right| \\
&\leq \left|\max_{z\sim P}\ell(\mathbf{M},\boldsymbol{\alpha},z)\sum_{i=1}^{M}\left|\frac{|N_i|}{d_l} - p(C_i)\right|\right| + \left|\frac{1}{d_l}\sum_{i=1}^{M}\sum_{j\in N_i}\max_{z\in C_i}|\ell(\mathbf{M},\boldsymbol{\alpha},z_j) - \ell(\mathbf{M},\boldsymbol{\alpha},z)|\right| \\
&\leq \frac{1}{\gamma}l\rho + B\sqrt{\frac{2M\ln 2 + 2\ln(1/\delta)}{d_l}}. \qquad \text{(B.2)}
\end{aligned}
$$

Inequality (B.1) is due to the triangle inequality. Inequality (B.2) comes from the application of Proposition 4.8 and Theorem 4.7. □

Properties of the similarity function $K^1_{\mathbf{M}}$. Recall that $K^1_{\mathbf{M}}(\mathbf{x},\mathbf{x}') = \mathbf{x}^T\mathbf{M}\mathbf{x}'$. In order to prove that $K^1_{\mathbf{M}}$ is 1-lipschitz, we need to bound the following difference.

$$
\begin{aligned}
|K^1_{\mathbf{M}}(\mathbf{x},\mathbf{x}'') - K^1_{\mathbf{M}}(\mathbf{x}',\mathbf{x}'')| &= \left|(\mathbf{x}^T\mathbf{M}\mathbf{x}'') - (\mathbf{x}'^T\mathbf{M}\mathbf{x}'')\right| \\
&= \left|(\mathbf{x}-\mathbf{x}')^T\mathbf{M}\mathbf{x}''\right| \\
&\leq ||\mathbf{x}-\mathbf{x}'||_2 \cdot ||\mathbf{M}\mathbf{x}''||_2 \qquad \text{(B.3)} \\
&\leq ||\mathbf{x}-\mathbf{x}'||. \qquad \text{(B.4)}
\end{aligned}
$$

Inequality (B.3) comes from the Cauchy-Schwarz inequality and some classical norm properties. Finally, Inequality (B.4) comes from Constraint (4.5) in JSL and the normalization of the data resulting in $||\mathbf{x}||_2 \leq 1$.

We now prove that $K^1_{\mathbf{M}}$ is (0,1)-admissible. It is straightforward to bound its absolute value:

$$
\begin{aligned}
|K^1_{\mathbf{M}}(\mathbf{x},\mathbf{x}')| &= |\mathbf{x}^T\mathbf{M}\mathbf{x}'| \\
&\leq \left\|\mathbf{x}'\mathbf{x}^T\right\|_2 \|\mathbf{M}\|_{\mathcal{F}} \qquad \text{(B.5)}
\end{aligned}
$$

We get Inequality (B.5) by applying the Cauchy-Schwarz inequality. □

Properties of the similarity function $K^2_{\mathbf{M}}$. Recall that $K^2_{\mathbf{M}}(\mathbf{x},\mathbf{x}') = 1-(\mathbf{x}-\mathbf{x}')^T\mathbf{M}(\mathbf{x}-\mathbf{x}')$. As in the case of similarity $K^1_{\mathbf{M}}$, we bound the following difference:

$$
\begin{aligned}
\left|K^2_{\mathbf{M}}(\mathbf{x},\mathbf{x}'') - K^2_{\mathbf{M}}(\mathbf{x}',\mathbf{x}'')\right| &= \left|1-(\mathbf{x}-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}-\mathbf{x}'') - 1 + (\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}'-\mathbf{x}'')\right| \\
&= \left|(\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}'-\mathbf{x}'') - (\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}-\mathbf{x}'')\right. \\
&\quad \left. + (\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}-\mathbf{x}'') - (\mathbf{x}-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}-\mathbf{x}'')\right| \\
&= \left|(\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}'-\mathbf{x}) + (\mathbf{x}'-\mathbf{x})^T\mathbf{M}(\mathbf{x}-\mathbf{x}'')\right| \\
&\leq \left|(\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}'-\mathbf{x})\right| + \left|(\mathbf{x}'-\mathbf{x})^T\mathbf{M}(\mathbf{x}-\mathbf{x}'')\right| \\
&\leq ||\mathbf{x}'-\mathbf{x}''||_2 \cdot ||\mathbf{M}\mathbf{x}'-\mathbf{M}\mathbf{x}||_2 + ||\mathbf{x}'-\mathbf{x}||_2 \cdot ||\mathbf{M}\mathbf{x}-\mathbf{M}\mathbf{x}''||_2 \qquad \text{(B.6)} \\
&\leq ||\mathbf{x}'-\mathbf{x}''||_2 \cdot (||\mathbf{x}'||_2 + ||\mathbf{x}||_2) + ||\mathbf{x}'-\mathbf{x}||_2 \cdot (||\mathbf{x}||_2 + ||\mathbf{x}''||_2) \qquad \text{(B.7)} \\
&\leq 4||\mathbf{x}-\mathbf{x}'||. \qquad \text{(B.8)}
\end{aligned}
$$

Inequalities (B.6) comes from the Cauchy-Schwarz inequality and some classical norm properties; Inequality (B.7) is due to Constraint (4.5) in JSL, and Inequality (B.8) is due to the normalization of the data $||\mathbf{x}||_2 \leq 1$.

We now prove that $K^2_{\mathbf{M}}$ is (1,4)-admissible.

$$
\begin{aligned}
|K^2_{\mathbf{M}}(\mathbf{x},\mathbf{x}')| &= |1-(\mathbf{x}-\mathbf{x}')^T\mathbf{M}(\mathbf{x}-\mathbf{x}')| \\
&\leq 1 + \left\|(\mathbf{x}-\mathbf{x}')^T\mathbf{M}(\mathbf{x}-\mathbf{x}')\right\| \\
&\leq 1 + \left\|\mathbf{x}-\mathbf{x}'\right\|_2^2 \|\mathbf{M}\|_{\mathcal{F}} \qquad \text{(B.9)}
\end{aligned}
$$

$$\leq 1 + 4 \left\|\mathbf{x}'\mathbf{x}^T\right\|_2 \|\mathbf{M}\|_{\mathcal{F}} \tag{B.10}$$

We get Inequality (B.9) by applying the Cauchy-Schwarz inequality. As the examples are normalized $\|\mathbf{x}\|_2 \leq 1$, we obtain Inequality (B.10). □

Properties of the similarity function $K^3_{\mathbf{M}}$. Recall that $K^3_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{(\mathbf{x}-\mathbf{x}')^T\mathbf{M}(\mathbf{x}-\mathbf{x}')}{2\sigma^2}\right)$. This proof is similar to the one for the previous similarities. We bound the following difference:

$$\begin{aligned}
&\left|K^3_{\mathbf{M}}(\mathbf{x}, \mathbf{x}'') - K^3_{\mathbf{M}}(\mathbf{x}', \mathbf{x}'')\right| = \left|\exp\left(-\frac{(\mathbf{x}-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}-\mathbf{x}'')}{2\sigma^2}\right) - \exp\left(-\frac{(\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}'-\mathbf{x}'')}{2\sigma^2}\right)\right| \\
\leq &\left(\exp\left(\frac{1}{2\sigma^2}\right) - \exp\left(\frac{-1}{2\sigma^2}\right)\right) \cdot \left|\frac{(\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}'-\mathbf{x}'')}{2\sigma^2} - \frac{(\mathbf{x}-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}-\mathbf{x}'')}{2\sigma^2}\right| \qquad \text{(B.11)} \\
= &\frac{1}{2\sigma^2}\left(\exp\left(\frac{1}{2\sigma^2}\right) - \exp\left(\frac{-1}{2\sigma^2}\right)\right) \\
&\cdot \left|(\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}'-\mathbf{x}'') - (\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}-\mathbf{x}'') + (\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}-\mathbf{x}'') - (\mathbf{x}-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}-\mathbf{x}'')\right| \\
= &\frac{1}{2\sigma^2}\left(\exp\left(\frac{1}{2\sigma^2}\right) - \exp\left(\frac{-1}{2\sigma^2}\right)\right) \cdot \left|(\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}'-\mathbf{x}) + (\mathbf{x}'-\mathbf{x})^T\mathbf{M}(\mathbf{x}-\mathbf{x}'')\right| \\
\leq &\frac{1}{2\sigma^2}\left(\exp\left(\frac{1}{2\sigma^2}\right) - \exp\left(\frac{-1}{2\sigma^2}\right)\right) \cdot \left(\left|(\mathbf{x}'-\mathbf{x}'')^T\mathbf{M}(\mathbf{x}'-\mathbf{x})\right| + \left|(\mathbf{x}'-\mathbf{x})^T\mathbf{M}(\mathbf{x}-\mathbf{x}'')\right|\right) \\
\leq &\frac{1}{2\sigma^2}\left(\exp\left(\frac{1}{2\sigma^2}\right) - \exp\left(\frac{-1}{2\sigma^2}\right)\right) \cdot (\|\mathbf{x}'-\mathbf{x}''\|_2 \cdot ||\mathbf{M}\mathbf{x}'-\mathbf{M}\mathbf{x}||_2 + \|\mathbf{x}'-\mathbf{x}\|_2 \cdot \|\mathbf{M}\mathbf{x}-\mathbf{M}\mathbf{x}''\|_2) \\
&\qquad \text{(B.12)} \\
\leq &\frac{1}{2\sigma^2}\left(\exp\left(\frac{1}{2\sigma^2}\right) - \exp\left(\frac{-1}{2\sigma^2}\right)\right) \\
&\cdot (\|\mathbf{x}'-\mathbf{x}''\|_2 \cdot (\|\mathbf{M}\mathbf{x}'\|_2 + \|\mathbf{M}\mathbf{x}\|_2) + \|\mathbf{x}'-\mathbf{x}\|_2 \cdot (\|\mathbf{M}\mathbf{x}\|_2 + \|\mathbf{M}\mathbf{x}''\|_2)) \qquad \text{(B.13)} \\
\leq &\frac{1}{2\sigma^2}\left(\exp\left(\frac{1}{2\sigma^2}\right) - \exp\left(\frac{-1}{2\sigma^2}\right)\right) \cdot (\|\mathbf{x}'-\mathbf{x}''\|_2 \cdot (\|\mathbf{x}'\|_2 + \|\mathbf{x}\|_2) + \|\mathbf{x}'-\mathbf{x}\|_2 \cdot (\|\mathbf{x}\|_2 + \|\mathbf{x}''\|_2)) \\
&\qquad \text{(B.14)} \\
\leq &\frac{1}{2\sigma^2}\left(\exp\left(\frac{1}{2\sigma^2}\right) - \exp\left(\frac{-1}{2\sigma^2}\right)\right) 4\|\mathbf{x}'-\mathbf{x}\| \qquad \text{(B.15)} \\
= &\frac{2}{\sigma^2}\left(\exp\left(\frac{1}{2\sigma^2}\right) - \exp\left(\frac{-1}{2\sigma^2}\right)\right) \|\mathbf{x}-\mathbf{x}'\|.
\end{aligned}$$

Inequality (B.11) is due to the l-lipschitzness of the exponential function on the range $[\frac{-1}{2\sigma^2}, \frac{1}{2\sigma^2}]$. Inequalities (B.12) and (B.13) come from the Cauchy-Schwarz inequality and some classical norm properties. Inequality (B.14) is due to Constraint (4.5) in JSL and Inequality (B.15) is due to $||\mathbf{x}||_2 \leq 1$.

We now prove that $K^3_{\mathbf{M}}$ is $\left(\exp(-2/\sigma^2), 0\right)$-admissible:

$$\begin{aligned}
|K^3_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')| &= \left|\exp\left(-\frac{(\mathbf{x}-\mathbf{x}')^T\mathbf{M}(\mathbf{x}-\mathbf{x}')}{2\sigma^2}\right)\right| \\
&\leq \exp\left(\frac{-4\left\|\mathbf{x}^T\mathbf{x}'\right\|_2 \|\mathbf{M}\|_{\mathcal{F}}}{2\sigma^2}\right) \qquad \text{(B.16)}
\end{aligned}$$

$$\leq \exp\left(-\frac{2}{\sigma^2}\right) \tag{B.17}$$

We use the $(1,4)$-admissibility of $K^2_{\mathbf{M}}$ to obtain Inequality (B.16), while Inequality (B.17) comes from the data normalization and the fact that $\|\mathbf{M}\|_{\mathcal{F}} \leq 1$. Here, we are only able to bound $K^3_{\mathbf{M}}$ by a constant. □

B.2 Proofs of Chapter 5

This section contains the proofs of Lemmas 5.1 to 5.6 from Chapter 5, as well as defining some additional lemmas necessary for these proofs.

To prove Lemma 5.1, we need two additional lemmas. Lemma B.1 bounds the Frobenius norm of the learned matrix $\mathbf{M}$, while Lemma B.2 puts a bound on the Frobenius norm of a subpart of the similarity function.

Lemma B.1. *If* $\mathbf{M}$ *is the optimal solution of Problem* (4.3)*, we have:*

$$||\mathbf{M}||_{\mathcal{F}} \leq \frac{1}{\sqrt{\lambda}}.$$

Proof. Since $\mathbf{M}$ is the optimal solution of Problem (4.3), we have:

$$\begin{aligned}
R_{\mathcal{S}}(\mathbf{M}) &\leq R_{\mathcal{S}}(\mathbf{0}) \\
\frac{1}{m}\sum_{(\mathbf{A},y)\in\mathcal{S}} \ell(\mathbf{M},(\mathbf{A},y)) + \lambda||\mathbf{M}||^2_{\mathcal{F}} &\leq \frac{1}{m}\sum_{(\mathbf{A},y)\in\mathcal{S}} \ell(\mathbf{0},(\mathbf{A},y)) + \lambda||\mathbf{0}||^2_{\mathcal{F}} \\
\lambda\,\|\mathbf{M}\|^2_{\mathcal{F}} &\leq \frac{1}{m}\sum_{(\mathbf{A},y)\in\mathcal{S}} \ell(\mathbf{0},(\mathbf{A},y)) \qquad \text{(B.18)} \\
\lambda\,\|\mathbf{M}\|^2_{\mathcal{F}} &\leq 1 \qquad \text{(B.19)} \\
\|\mathbf{M}\|_{\mathcal{F}} &\leq \frac{1}{\sqrt{\lambda}}
\end{aligned}$$

Inequality (B.18) is a result of the fact that the hinge loss is always positive, while Inequality (B.19) comes from noting that the loss is bounded by $1/m$ when the metric is set to zero. □

Lemma B.2 (Technical lemma)**.** *Let* $\mathbf{A} \in \mathbb{R}^{t_{\mathbf{A}}\times d}$ *and* $\mathbf{B} \in \mathbb{R}^{t_{\mathbf{B}}\times d}$ *be two examples, and* $\mathbf{Y_{AB}} \in \{0,1\}^{t_{\mathbf{A}}\times t_{\mathbf{B}}}$ *of length* $t_{\mathbf{AB}}$*. Then*

$$\left\|\mathbf{A}^T \cdot \mathbf{Y_{AB}} \cdot \mathbf{B}\right\|_{\mathcal{F}} \leq t_{\mathbf{AB}}\sqrt{2d}.$$

Proof.

$$\left\|\mathbf{A}^T \cdot \mathbf{Y_{AB}} \cdot \mathbf{B}\right\|_{\mathcal{F}}$$

$$
\begin{aligned}
&= \sqrt{\sum_{i=1}^{d}\sum_{t=1}^{d}\left(\sum_{j=1}^{t_{\mathbf{A}}}\sum_{k=1}^{t_{\mathbf{B}}} a_{ij}y_{jk}b_{kt}\right)^2} \\
&= \sqrt{\sum_{i=1}^{d}\sum_{t=1}^{d}\left(2\sum_{j=1}^{t_{\mathbf{A}}}\sum_{k=1}^{t_{\mathbf{B}}}\sum_{j'=1}^{t_{\mathbf{A}}}\sum_{k'=1}^{t_{\mathbf{B}}}(a_{ij}y_{jk}b_{kt})(a_{ij'}y_{j'k'}b_{k't}) - \sum_{jk}(a_{ij}y_{jk}b_{kt})^2\right)} \\
&\leq \sqrt{\sum_{i,t} 2\sum_{j,k}\sum_{j',k'} |(a_{ij}y_{jk}b_{kt})(a_{ij'}y_{j'k'}b_{k't})|} \\
&= \sqrt{2\sum_{j,k} y_{jk}\sum_{j',k'} y_{j'k'} + \sum_{i,t} a_{ij}a_{ij'}b_{kt}b_{k't}} \\
&\leq \sqrt{2t_{\mathbf{AB}}^2 |\sum_i a_{ij}a_{ij'}\sum_t b_{kt}b_{k't}|} \\
&= \sqrt{2t_{\mathbf{AB}}^2 \max_i a_{ij}\sum_i |a_{ij}| \max_t b_{kt}\sum_t |b_{kt}|} \\
&= \sqrt{2t_{\mathbf{AB}}^2 \max_i a_{ij}\,\|\mathbf{a}_i\|_1 \max_t b_{kt}\,\|\mathbf{b}_k\|_1} \\
&\leq \sqrt{2t_{\mathbf{AB}}^2\sqrt{d}\,\|\mathbf{a}_i\|_2\sqrt{d}\,\|\mathbf{b}_k\|_2} \\
&\leq \sqrt{2t_{\mathbf{AB}}^2\cdot d} \\
&= t_{\mathbf{AB}}\sqrt{2d}.
\end{aligned}
$$

□

We are now able to present the proof of Lemma 5.1:

Proof of Lemma 5.1.

$$
\begin{aligned}
\ell(\mathbf{M},(\mathbf{A},y)) &= \left[1 - \frac{1}{n}\sum_{(\mathbf{B},y')\in\mathcal{L}} yy'\,\mathrm{K}_{\mathbf{M}}(\mathbf{A},\mathbf{B})/\gamma\right]_+ \\
&\leq \left|y\frac{1}{n}\sum_{(\mathbf{B},y')\in\mathcal{L}} y'\,\mathrm{K}_{\mathbf{M}}(\mathbf{A},\mathbf{B})/\gamma\right| && \text{(B.20)} \\
&\leq \frac{1}{n}\sum_{(\mathbf{B},y')\in\mathcal{L}} \left|y'\,\mathrm{K}_{\mathbf{M}}(\mathbf{A},\mathbf{B})/\gamma\right| && \text{(B.21)} \\
&\leq \frac{1}{n\gamma}\sum_{(\mathbf{B},y')\in\mathcal{L}} \left|\mathrm{tr}(\mathbf{M}^T\mathbf{A}^T\mathbf{Y}_{\mathbf{AB}}\mathbf{B})/t_{\mathbf{AB}}\right| \\
&\leq \frac{1}{n\gamma}\sum_{(\mathbf{B},y')\in\mathcal{L}} \frac{1}{t_{\mathbf{AB}}}\|\mathbf{M}\|_{\mathcal{F}}\left\|\mathbf{A}^T\mathbf{Y}_{\mathbf{AB}}\mathbf{B}\right\|_{\mathcal{F}} \\
&\leq \frac{1}{n\gamma}\sum_{(\mathbf{B},y')\in\mathcal{L}} \frac{1}{t_{\mathbf{AB}}}\frac{1}{\sqrt{\lambda}}t_{\mathbf{AB}}\sqrt{2d} && \text{(B.22)} \\
&\leq \frac{\sqrt{2d}}{\gamma\sqrt{\lambda}}.
\end{aligned}
$$

Equation (B.20) comes from the 1-lipschitzness of the hinge loss. Inequality (B.21) is obtained by applying triangle inequality. We obtain line (B.22) by applying Lemmas B.1 and B.2. □

Proof of Lemma 5.2.

$$
\begin{aligned}
&|\ell(\mathbf{M},(\mathbf{A},y)) - \ell(\mathbf{M}',(\mathbf{A},y))| \\
=&\left|\left[1-\frac{1}{n}\sum_{(\mathbf{B},y')\in\mathcal{L}} ll'\,\mathrm{K}_{\mathbf{M}}(\mathbf{A},\mathbf{B})/\gamma\right]_+ - \left[1-\frac{1}{n}\sum_{(\mathbf{B},y')\in\mathcal{L}} ll'\,\mathrm{K}_{\mathbf{M}'}(\mathbf{A},\mathbf{B})/\gamma\right]_+\right| \\
\leq&\left|\frac{1}{n}\sum_{(\mathbf{B},y')\in\mathcal{L}} ll'\,\mathrm{K}_{\mathbf{M}}(\mathbf{A},\mathbf{B})/\gamma - \frac{1}{n}\sum_{(\mathbf{B},y')\in\mathcal{L}} ll'\,\mathrm{K}_{\mathbf{M}'}(\mathbf{A},\mathbf{B})/\gamma\right| \qquad \text{(B.23)} \\
=&\frac{1}{n\gamma}\left|\sum_{(\mathbf{B},y')\in\mathcal{L}} y'\left(\mathrm{K}_{\mathbf{M}}(\mathbf{A},\mathbf{B}) - \mathrm{K}_{\mathbf{M}'}(\mathbf{A},\mathbf{B})\right)\right| \\
\leq&\frac{1}{n\gamma}\sum_{(\mathbf{B},y')\in\mathcal{L}} \left|\mathrm{K}_{\mathbf{M}}(\mathbf{A},\mathbf{B}) - \mathrm{K}_{\mathbf{M}'}(\mathbf{A},\mathbf{B})\right| \qquad \text{(B.24)} \\
=&\frac{1}{n\gamma}\sum_{(\mathbf{B},y')\in\mathcal{L}} \left|\mathrm{tr}((\mathbf{M}-\mathbf{M}')^T\cdot\mathbf{A}^T\cdot\mathbf{Y}_{\mathbf{AB}}\cdot\mathbf{B})/t_{\mathbf{AB}}\right| \\
\leq&\frac{1}{n\gamma}\sum_{(\mathbf{B},y')\in\mathcal{L}} \frac{1}{t_{\mathbf{AB}}}||(\mathbf{M}-\mathbf{M}')^T\cdot\mathbf{A}^T\cdot\mathbf{Y}_{\mathbf{AB}}\cdot\mathbf{B})||_1 \qquad \text{(B.25)} \\
\leq&\frac{1}{n\gamma}\left\|\mathbf{M}-\mathbf{M}'\right\|_{\mathcal{F}}\sum_{(\mathbf{B},y')\in\mathcal{L}} \frac{1}{t_{\mathbf{AB}}}\left\|\mathbf{A}^T\cdot\mathbf{Y}_{\mathbf{AB}}\cdot\mathbf{B}\right\|_{\mathcal{F}} \qquad \text{(B.26)} \\
\leq&\frac{\sqrt{2d}}{\gamma}\left\|\mathbf{M}-\mathbf{M}'\right\|_{\mathcal{F}}. \qquad \text{(B.27)}
\end{aligned}
$$

Inequality (B.23) comes from the 1-lipschitzness of the hinge loss. Inequality (B.24) is obtained by applying triangle inequality. By using Lemma B.2 on line (B.26), we obtain the lemma. □

Proof of Lemma 5.3. This proof is similar to the one of Lemma 20 in Bousquet & Elisseeff (2002). Consider the following notation for the objective function of SLTS (Equation (4.3)):

$$F_{\mathcal{S}}(\mathbf{M}) := R_{\mathcal{S}}(\mathbf{M}) + \lambda\left\|\mathbf{M}\right\|_{\mathcal{F}}^2.$$

$R_{\mathcal{S}}(\cdot)$ is a convex function, thus for all $t\in[0,1]$, we have:

$$
\begin{aligned}
R_{\mathcal{S}^i}(\mathbf{M}-t\Delta\mathbf{M}) - R_{\mathcal{S}^i}(\mathbf{M}) &\leq t(R_{\mathcal{S}^i}(\mathbf{M}^i) - R_{\mathcal{S}^i}(\mathbf{M})), \qquad \text{(B.28)}\\
R_{\mathcal{S}^i}(\mathbf{M}^i+t\Delta\mathbf{M}) - R_{\mathcal{S}^i}(\mathbf{M}^i) &\leq t(R_{\mathcal{S}^i}(\mathbf{M}) - R_{\mathcal{S}^i}(\mathbf{M}^i)). \qquad \text{(B.29)}
\end{aligned}
$$

By summing Inequalities (B.28) and (B.29) we obtain the following:

$$R_{\mathcal{S}^i}(\mathbf{M}-t\Delta\mathbf{M}) - R_{\mathcal{S}^i}(\mathbf{M}) + R_{\mathcal{S}^i}(\mathbf{M}^i+t\Delta\mathbf{M}) - R_{\mathcal{S}^i}(\mathbf{M}^i) \leq 0. \qquad \text{(B.30)}$$

Since $\mathbf{M}$ and $\mathbf{M}^i$ are respectively the minimizers of $F_{\mathcal{S}}(\cdot)$ and $F_{\mathcal{S}^i}(\cdot)$, we can write:

$$R_{\mathcal{S}}(\mathbf{M}) - R_{\mathcal{S}}(\mathbf{M} - t\Delta\mathbf{M}) \leq 0, \tag{B.31}$$
$$R_{\mathcal{S}^i}(\mathbf{M}^i) - R_{\mathcal{S}^i}(\mathbf{M}^i - t\Delta\mathbf{M}) \leq 0. \tag{B.32}$$

By summing Inequalities (B.31) and (B.32) we obtain:

$$\begin{aligned} F_{\mathcal{S}}(\mathbf{M}) - F_{\mathcal{S}}(\mathbf{M} - t\Delta\mathbf{M}) + F_{\mathcal{S}^i}(\mathbf{M}^i) - F_{\mathcal{S}^i}(\mathbf{M}^i - t\Delta\mathbf{M}) \leq 0 \\ R_{\mathcal{S}}(\mathbf{M}) - R_{\mathcal{S}}(\mathbf{M} - t\Delta\mathbf{M}) + \lambda||\mathbf{M}||_{\mathcal{F}}^2 - \lambda||\mathbf{M} - t\Delta\mathbf{M}||_{\mathcal{F}}^2 \\ +R_{\mathcal{S}^i}(\mathbf{M}^i) - R_{\mathcal{S}^i}(\mathbf{M}^i + t\Delta\mathbf{M}) + \lambda||\mathbf{M}^i||_{\mathcal{F}}^2 - \lambda||\mathbf{M}^i + t\Delta\mathbf{M}||_{\mathcal{F}}^2 \leq 0 \end{aligned} \tag{B.33}$$

We can now sum Inequalities (B.30) and (B.33):

$$\begin{aligned} R_{\mathcal{S}}(\mathbf{M}) - R_{\mathcal{S}}(\mathbf{M} - t\Delta\mathbf{M}) + \lambda||\mathbf{M}||_{\mathcal{F}}^2 - \lambda||\mathbf{M} - t\Delta\mathbf{M}||_{\mathcal{F}}^2 \\ +R_{\mathcal{S}^i}(\mathbf{M}^i) - R_{\mathcal{S}^i}(\mathbf{M}^i + t\Delta\mathbf{M}) + \lambda \left\|\mathbf{M}^i\right\|_{\mathcal{F}}^2 - \lambda \left\|\mathbf{M}^i + t\Delta\mathbf{M}\right\|_{\mathcal{F}}^2 \leq 0 \end{aligned}$$

From the previous inequality, we can write:

$$\lambda \left\|\mathbf{M}\right\|_{\mathcal{F}}^2 - \lambda \left\|\mathbf{M} - t\Delta\mathbf{M}\right\|_{\mathcal{F}}^2 + \lambda \left\|\mathbf{M}^i\right\|_{\mathcal{F}}^2 - \lambda \left\|\mathbf{M}^i + t\Delta\mathbf{M}\right\|_{\mathcal{F}}^2 \leq Q, \tag{B.34}$$

with

$$Q = R_{\mathcal{S}^i}(\mathbf{M}) - R_{\mathcal{S}}(\mathbf{M}) + R_{\mathcal{S}}(\mathbf{M} - t\Delta\mathbf{M}) - R_{\mathcal{S}^i}(\mathbf{M} - t\Delta\mathbf{M}).$$

We now need to bound the previous quantity Q:

$$\begin{aligned} Q &\leq \frac{1}{m} \Bigg| \sum_{(\mathbf{A},y)\in\mathcal{S}^i} \ell(\mathbf{M}, (\mathbf{A}, y)) - \sum_{(\mathbf{A},y)\in\mathcal{S}} \ell(\mathbf{M}, (\mathbf{A}, y)) + \sum_{(\mathbf{A},y)\in\mathcal{S}} \ell(\mathbf{M} - t\Delta\mathbf{M}, (\mathbf{A}, y)) \\ &\quad - \sum_{(\mathbf{A},y)\in\mathcal{S}^i} \ell(\mathbf{M} - t\Delta\mathbf{M}, (\mathbf{A}, y)) \Bigg| \\ &= \frac{1}{m} \left| \ell(\mathbf{M}, (\mathbf{A}^i, y^i)) - \ell(\mathbf{M}, (\mathbf{A}, y)) + \ell(\mathbf{M} - t\Delta\mathbf{M}, (\mathbf{A}, y)) - \ell(\mathbf{M} - t\Delta\mathbf{M}, (\mathbf{A}^i, y^i)) \right| \end{aligned} \tag{B.35}$$

$$\leq \frac{1}{m} \left(\left| \ell(\mathbf{M}, (\mathbf{A}^i, y^i)) - \ell(\mathbf{M} - t\Delta\mathbf{M}, (\mathbf{A}^i, y^i)) \right| + \left| \ell(\mathbf{M} - t\Delta\mathbf{M}, (\mathbf{A}, y)) - \ell(\mathbf{M}, (\mathbf{A}, y)) \right| \right) \tag{B.36}$$

$$\leq \frac{1}{m} (l \left\|\mathbf{M} - \mathbf{M} + t\Delta\mathbf{M}\right\|_{\mathcal{F}} + l \left\|\mathbf{M} - t\Delta\mathbf{M} - \mathbf{M}\right\|_{\mathcal{F}}) \tag{B.37}$$

$$= \frac{2lt}{m} \left\|\Delta\mathbf{M}\right\|_{\mathcal{F}}$$

On line (B.35) we keep the terms that differ in the previous sums. We get Inequality (B.36) from triangle inequality, while (B.37) comes from the l-lipschitzness of the loss function (Lemma 5.2). Combining the bound on Q with Equation (B.34) proves the lemma. □

Proof of Lemma 5.5.

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}[\mathcal{E}_{\mathcal{S}}] &\leq \mathbb{E}_{\mathcal{S}}[\mathbb{E}_{(\mathbf{A},y)}[\ell(\mathbf{M},(\mathbf{A},y))] - R_{\mathcal{S}}(\mathbf{M})] \\
&\leq \mathbb{E}_{\mathcal{S},(\mathbf{A},y)}\left[\left|\ell(\mathbf{M},(\mathbf{A},y)) - \frac{1}{m}\sum_{(\mathbf{A}_i,y_i)\in\mathcal{S}}\ell(\mathbf{M},(\mathbf{A}_i,y_i))\right|\right] \\
&\leq \mathbb{E}_{\mathcal{S},(\mathbf{A},y)}\left[\left|\frac{1}{m}\sum_{(\mathbf{A}_i,y_i)}(\ell(\mathbf{M},(\mathbf{A},y)) - \ell(\mathbf{M},(\mathbf{A}_i,y_i)))\right|\right] \\
&\leq \mathbb{E}_{\mathcal{S},(\mathbf{A},y)}\left[\left|\frac{1}{m}\sum_{(\mathbf{A}_i,y_i)}\left(\ell(\mathbf{M}^i,(\mathbf{A}_i,y_i)) - \ell(\mathbf{M},(\mathbf{A}_i,y_i))\right)\right|\right] \qquad \text{(B.38)} \\
&\leq \frac{\kappa}{m}. \qquad \text{(B.39)}
\end{aligned}$$

Inequality (B.38) comes from the fact that changing one point with another from the same distribution does not affect the expected value, while Inequality (B.39) results from applying triangle inequality and uniform stability (Theorem 5.4). □

Proof of Lemma 5.6.

$$\begin{aligned}
&|\mathcal{E}_{\mathcal{S}} - \mathcal{E}_{\mathcal{S}^i}| \\
=&|R_P(\mathbf{M}) - R_{\mathcal{S}}(\mathbf{M}) - R_P(\mathbf{M}^i) + R_{\mathcal{S}^i}(\mathbf{M}^i)| \\
=&|R_P(\mathbf{M}) - R_{\mathcal{S}}(\mathbf{M}) - R_P(\mathbf{M}^i) + R_{\mathcal{S}^i}(\mathbf{M}^i) - R_{\mathcal{S}}(\mathbf{M}^i) + R_{\mathcal{S}}(\mathbf{M}^i)| \\
\leq&|R_P(\mathbf{M}) - R_P(\mathbf{M}^i)| + |R_{\mathcal{S}}(\mathbf{M}^i) - R_{\mathcal{S}}(\mathbf{M})| + |R_{\mathcal{S}^i}(\mathbf{M}^i) - R_{\mathcal{S}}(\mathbf{M}^i)| \qquad \text{(B.40)} \\
\leq&\mathbb{E}_{(\mathbf{A},y)}[|\ell(\mathbf{M},(\mathbf{A},y)) - \ell(\mathbf{M}^i,(\mathbf{A},y))|] + |R_{\mathcal{S}}(\mathbf{M}^i) - R_{\mathcal{S}}(\mathbf{M})| + |R_{\mathcal{S}^i}(\mathbf{M}^i) - R_{\mathcal{S}}(\mathbf{M}^i)| \\
& \qquad \text{(B.41)} \\
\leq&\frac{\kappa}{m} + |R_{\mathcal{S}}(\mathbf{M}^i) - R_{\mathcal{S}}(\mathbf{M})| + |R_{\mathcal{S}^i}(\mathbf{M}^i) - R_{\mathcal{S}}(\mathbf{M}^i)| \qquad \text{(B.42)} \\
\leq&\frac{\kappa}{m} + \frac{1}{m}\sum_{(\mathbf{A},y)\in\mathcal{S}}\left|\ell(\mathbf{M}^i,(\mathbf{A},y)) - \ell(\mathbf{M},(\mathbf{A},y))\right| \\
&+|R_{\mathcal{S}^i}(\mathbf{M}^i) - R_{\mathcal{S}}(\mathbf{M}^i)| \\
\leq&\frac{\kappa}{m} + \frac{\kappa}{m} + |R_{\mathcal{S}^i}(\mathbf{M}^i) - R_{\mathcal{S}}(\mathbf{M}^i)| \qquad \text{(B.43)} \\
=&\frac{2\kappa}{m} + \frac{1}{m}|\ell(\mathbf{M}^i,(\mathbf{A}^i,y^i)) - \ell(\mathbf{M}^i,(\mathbf{A},y))| \qquad \text{(B.44)} \\
\leq&\frac{2\kappa}{m} + \frac{1}{m}|\ell(\mathbf{M}^i,(\mathbf{A}^i,y^i))| \qquad \text{(B.45)} \\
\leq&\frac{2\kappa}{m} + \frac{\sqrt{2d}}{m\gamma\sqrt{\lambda}} \qquad \text{(B.46)}
\end{aligned}$$

Inequalities (B.40) and (B.41) come from triangle inequality. Inequalities (B.42) and (B.43) come from the uniform stability of our algorithm (Theorem 5.4). Line (B.44) comes from the fact that $\mathcal{S}$ and $\mathcal{S}^i$ differ only by example i. We can write Inequality (B.45) because the loss is always positive, and we get line (B.46) by bounding the value of the loss function (Lemma 5.1). □

Appendix C

Translation into French

Table des matières

Introduction

L'apprentissage automatique a pour but de concevoir des algorithmes de prédiction précis à partir des données d'entraînement. Ces données peuvent être vues comme une forme d'expérience utilisée pour de l'apprentissage par l'exemple : les nouvelles décisions prennent en compte les instances précédentes. Ce processus est différent des systèmes de mémorisation, qui reconnaissant uniquement les exemples déjà rencontrés. Dans l'apprentissage, le but est de déterminer le modèle associé aux données et d'être capable de l'appliquer sur des nouvelles instances. Dans la plupart des cas, une fois l'étape de modélisation effectuée, les données initiales peuvent être ignorées et le modèle peut prédire les résultats pour des nouvelles instances arrivant dans le système. Contrairement aux systèmes basés sur la mémoire, cela implique que l'apprentissage automatique a des capacités de généralisation.

Les algorithmes d'apprentissage sont utilisés avec succès dans une variété d'applications, y compris :

- La vision par ordinateur (par exemple la segmentation d'image, la détection de visage ou bien la reconnaissance d'objet) ;
- Le traitement du signal (par exemple la reconnaissance et synthèse de la parole ou l'identification vocale) ;
- La recherche d'information (par exemple les moteurs de recherche ou les systèmes de recommandation) ;
- Les véhicules et robots autonomes (par exemple les drones ou les voitures auto-dirigées) ;
- La linguistique informatique ;
- La biologie et la génétique ;
- Le diagnostic médical.

Cette liste n'est pas exhaustive et de nouvelles applications pour les algorithmes d'apprentissage sont proposées chaque jour.

Les données peuvent être mises à la disposition de l'algorithme sous différentes formes. Les vecteurs de *features* sont souvent utilisés pour stocker des informations catégorielles ou numériques, comme l'emplacement d'une maison, son prix ou sa surface. Les données structurées sont utilisées pour représenter des informations qui ne peuvent pas être stockées directement en tant que vecteurs de *features*, telles que des chaînes (par exemple, des documents textuels), des arbres (par exemple, du contenu XML) ou des graphes (par exemple, des réseaux sociaux). Un autre type de données structurées présent dans de nombreuses applications sont les séries temporelles qui suivent l'évolution d'un processus à travers le temps (par exemple, les cotations boursières, les signes vitaux des patients ou les données météorologiques). Depuis un type de données particulier, ou une combinaison de plusieurs types, l'algorithme peut effectuer différentes tâches d'apprentissage.

En apprentissage supervisé, les données possèdent une étiquette associée à chaque instance. L'algorithme doit apprendre à prédire ces étiquettes pour de nouveaux exemples. Lorsque l'étiquette provient d'un ensemble discret de valeurs (par exemple, le genre d'une personne), la tâche est appelée classification. Inversement, lorsque les étiquettes sont continues (par exemple, le prix), la tâche à effectuer est une régression. Dans ce document, nous nous concentrons principalement sur les tâches supervisées. L'apprentissage non supervisé couvre le cas où aucune information d'étiquette n'est disponible pour les données. Dans ce contexte, l'algorithme est amené à exécuter une tâche pour laquelle l'information de l'étiquette n'est pas nécessaire. Par exemple, le *clustering* vise à trouver de la structure dans les données permettant de diviser les exemples en groupes nommés *clusters*. L'apprentissage semi-supervisé couvre des tâches similaires à celles du cadre supervisé, mais il permet d'intégrer en outre des échantillons non étiquetés.

La performance d'un algorithme d'apprentissage est fortement liée à la quantité et à la qualité des données disponibles. Traditionnellement, la performance d'un algorithme est associée à sa complexité algorithmique et sa consommation en mémoire. Bien que l'efficacité de calcul soit aussi une préoccupation importante dans l'apprentissage automatique, dans ce cas la performance est mesurée principalement par rapport à la qualité de la prédiction. Une autre mesure à prendre en compte est la complexité de l'échantillon, qui indique la quantité de données requises par un algorithme pour apprendre à résoudre une certaine tâche. Plus précisément, des garanties théoriques peuvent être fournies pour la performance d'un algorithme d'apprentissage par rapport à la quantité de données disponibles et une notion de complexité du modèle choisi. Ces types de garanties peuvent être dérivés en utilisant des méthodes de la théorie de l'apprentissage statistique.

La plupart des algorithmes d'apprentissage automatique sont amenés à comparer des instances de données, soit pour construire le modèle, soit pour faire des prédictions. Les comparaisons sont effectuées en utilisant une notion de métrique qui donne une mesure de ressemblance entre les exemples. Ceci est vrai pour les méthodes supervisées, comme les k plus proches voisins ou les machines à vecteurs de support, ainsi que pour les non-supervisés, tels que K-moyennes. Cependant, le choix d'une métrique adaptée peut s'avérer une tâche difficile et la performance de l'algorithme en dépend fortement. On cherche à choisir une mesure qui discrimine effectivement les données pour la tâche à accomplir (par exemple, pour la classification, en reconnaissant les instances d'une même étiquette comme étant similaires). De plus, les métriques standards, telles que la distance euclidienne, n'ont pas la capacité de s'adapter à une tâche ou d'intégrer des contraintes sémantiques, parfois ignorant l'information pertinente.

L'apprentissage de métriques vise à résoudre ce problème en apprenant automatiquement des métriques adaptées aux données. Cette étape est effectuée en amont de l'application d'un algorithme d'apprentissage automatique. La métrique précédemment acquise est utilisée par la suite pour résoudre une tâche d'apprentissage spécifique. L'apprentissage de métriques destiné à la classification est le sujet principal que nous abordons dans cette thèse. Ce domaine peut être considéré comme faisant partie d'un ensemble plus vaste appelé l'apprentissage de représentation : sous la nouvelle métrique, la représentation des données change. L'apprentissage de représentation comprend également des domaines tels

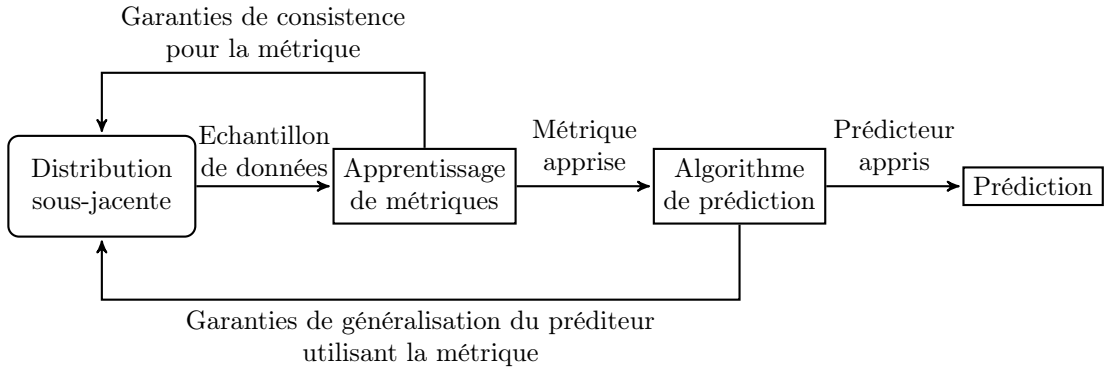


Figure C.1: Les étapes de l'apprentissage supervisé. La capacité de généralisation peut être évaluée à deux niveaux: pour la métrique et pour le prédicteur qui l'utilise.

que l'apprentissage de *features*, le *deep learning*, l'apprentissage de noyaux et bien d'autres. L'apprentissage de métriques a reçu beaucoup d'attention au cours des quinze dernières années, en particulier pour le cadre supervisé. La plupart des méthodes apprennent les paramètres d'une distance de Mahalanobis sous des contraintes induites par les données, ce qui équivaut à changer la représentation des instances sous une transformation linéaire. Dans le cas supervisé, la métrique apprise a pour but d'illustrer les instances avec les mêmes étiquettes comme similaires et les différentes comme dissemblables. Les méthodes de l'état de l'art dédiées aux vecteurs de *features* sont plus aptes à passer à l'echelle et plus efficaces que les approches précédentes. D'autre part, l'apprentissage de métriques pour les séries temporelles a reçu peu d'attention, probablement à cause de la structure plus complexe des données. Les séries temporelles issues des applications réelles ont généralement des longueurs, des taux d'échantillonnage et des phases différents. C'est pourquoi il est nécessaire d'aligner deux séries temporelles afin de les comparer. Cela implique de trouver toutes les correspondances entre les moments de temps. La déformation temporelle dynamique (DTW, pour *dynamic time warping* en anglais) est l'algorithme le plus connu pour trouver l'alignement optimal basé sur une matrice de coût. L'apprentissage de métriques pour les séries temporelles vise généralement à optimiser ce coût sous des contraintes imposées par les exemples.

L'état d'avancement en apprentissage de métriques souffre de quelques limitations non négligeables. La plus importante est probablement le manque de garanties théoriques pour la plupart des méthodes existantes. En effet, on voudrait évaluer la capacité de généralisation associée à l'apprentissage d'une métrique, qui se décline sous deux aspects (voir la Figure C.1). Premièrement, la métrique doit être consistente avec les données. Ce type de garantie n'a été fourni que pour un faible nombre de méthodes. Deuxièmement, la métrique apprise doit être liée à l'algorithme qui l'utilise, améliorant ses performances par rapport aux métriques standard. Par exemple, une métrique apprise à partir de contraintes locales pour une règle de classification locale peut ne pas fonctionner correctement lorsqu'elle est utilisée avec un classifieur linéaire global. Ce second type de garantie est aussi rare dans le domaine et, même si certaines méthodes améliorent un critère relatif au classifieur, ce lien n'est pas formellement établi. En pratique, la métrique apprise est utilisée dans le

classifieur dans l'espoir d'améliorer sa performance. La théorie des fonctions de similarité (ϵ, γ, τ)-bonnes a été l'un des premiers résultats à lier les propriétés d'une fonction de similarité à la performance qu'on peut attendre d'elle en classification. Nous décrivons ce cadre en détail dans la thèse, nos contributions en faisant usage. Une deuxième limitation dans l'apprentissage de métriques vient du fait que la plupart des méthodes fonctionnent avec des métriques qui imposent des propriétés de distance. Satisfaire ce type de contraintes peut être coûteux même en utilisant des techniques modernes d'optimisation numérique, ce qui rajoute un coût supplementaire à l'apprentissage. Cette limitation jointe à l'absence de garanties pour l'amélioration de la performance implique que l'effort dépensé dans l'apprentissage de métriques pourrait effectivement se traduire par une dégradation de la performance.

Cette thèse a pour but d'aborder les limitations mentionnées précédemment. Toutes nos contributions sont basées sur l'apprentissage de fonctions de similarité, une famille de métriques différente qui sont moins contraignantes que les distances et constituent une voie prometteuse mais moins explorée pour la recherche. Nous apportons également des garanties de généralisation pour la métrique apprise et le classifieur associé pour toutes les méthodes que nous proposons. Dans un premier temps, nous présentons un cadre général permettant d'apprendre simultanément la similarité et le classifieur linéaire. Ce cadre est conçu pour les vecteurs de *features* et est applicable pour une large gamme de fonctions de similarité. De plus, l'approche est semi-supervisée, ce qui lui permet d'exploiter des exemples non-étiquettés afin d'améliorer les performances. Notre contribution suivante nous permet d'aborder le problème de l'apprentissage de métriques pour les séries temporelles multivariées. Nous proposons d'apprendre une fonction de similarité bilinéaire destinée à la classification linéaire. La similarité utilise les alignements optimaux et permet de mieux repondérer les *features*.

Contexte de la thèse Les travaux de cette thèse ont été réalisés dans des équipes d'apprentissage automatique de deux établissements: le groupe Data Intelligence du Laboratoire Hubert Curien UMR CNRS 5516, de l'Université de Saint-Étienne et de l'Université de Lyon, et l'équipe Data Analysis, Modeling and Machine Learning (AMA) du Laboratoire d'Informatique de Grenoble, de l'Université Grenoble-Alpes. Le financement de ce projet a été assuré par une bourse de la Région Rhône-Alpes.

Résumé de la thèse Cette thèse est organisée comme suit.

- Le Chapitre 2 présente les notions relatives à l'apprentissage statistique qui sont nécessaires pour le reste de ce document. Nous commençons par une présentation formelle de l'apprentissage supervisé, puis donnons quelques exemples d'algorithmes d'apprentissage classiques, suivis des cadres théoriques de l'apprentissage pour obtenir des garanties de généralisation.
- Le Chapitre 3 est consacré aux métriques en général et à l'apprentissage de métriques en particulier. Après avoir présenté un certain nombre de mesures standard pour les vecteurs de *features* et les séries temporelles, nous fournissons une étude de

l'apprentissage de métriques (semi)-supervisé pour ces types de données. Ce chapitre explique plus en détail les limites qui ont déterminé l'orientation de nos contributions.

- Le Chapitre 4 présente nos contributions sur l'apprentissage des métriques pour les vecteurs de *features*. Nous introduisons un cadre général appelé JSL (pour *Joint Similarity Learning* en anglais) pour apprendre simultanément une fonction de similarité et un classifieur linéaire global. JSL est capable d'intégrer des contraintes globales basées sur des exemples non-étiquettées que l'on appelle des *landmarks*. Cette propriété permet à notre cadre de s'adapter au contexte particulier dans lequel seulement une petite quantité de données étiquetées est disponible. La formulation optimise les paramètres du cadre (ϵ, γ, τ)-bon de la similarité, ce qui assure sa performance en classification. JSL est convexe et peut être résolu efficacement sous de légères contraintes sur la fonction de similarité. Nous dérivons des garanties théoriques pour JSL en utilisant deux cadres différents: la robustesse algorithmique et la convergence uniforme basée sur la complexité de Rademacher. Les expériences comparent JSL à une vaste gamme de méthodes de l'état de l'art et prouvent son efficacité.

- Le Chapitre 5 rassemble nos contributions sur l'apprentissage de métriques pour les données temporelles en introduisant SLTS (pour *Similarity Learning for Time Series* en anglais). Nous proposons d'apprendre une fonction de similarité bilinéaire pour des séries temporelles multivariées basée sur les alignements optimaux. La similarité apprise est utilisée par la suite pour la classification linéaire. Le problème que nous résolvons est convexe et facile à résoudre. En utilisant le cadre de la stabilité uniforme, nous fournissons les premières garanties théoriques sous la forme d'une borne de généralisation pour la similarité apprise. De plus, en optimisant le critère du cadre (ϵ, γ, τ)-bon de la métrique, nous fournissons des garanties concernant sa performance pour la classification linéaire. L'étude expérimentale montre que l'approche proposée est efficace, tout en fournissant des classifieurs parcimonieux.

- Le Chapitre 6 conclut notre travail et discute des pistes pour des travaux futurs possibles.

Notations Le Tableau C.1 explique les notations utilisées tout au long de cette thèse.

Résumé des chapitres

Chapitre 2 Ce chapitre présente les configurations d'apprentissage les plus courantes, en se concentrant sur le cadre standard pour l'apprentissage supervisé, ainsi que certaines notions fondamentales de la théorie de l'apprentissage statistique. Nous présentons les principaux cadres analytiques et les outils permettant de dégager des garanties de généralisation. Nous commençons par le modèle d'apprentissage *Probably Approximately Correct* (PAC), puis nous présentons plusieurs cadres pour dériver des bornes de généralisation PAC : la convergence uniforme avec différentes mesures de complexité, la stabilité uniforme et la robustesse algorithmique.

Notation	Description
$\mathbb{R}$	L'ensemble des nombres réels
$\mathbb{R}^d$	L'ensemble des vecteurs réels de dimension d
$\mathbb{R}^{d \times d'}$	L'ensemble de matrices réelles de dimensions $d \times d'$
$\mathbb{S}_+^d$	Le cône des matrices réelles symétriques PSD de dimensions $d \times d$
$\mathcal{S}$	Un ensemble arbitraire
$\lvert\mathcal{S}\rvert$	Le cardinal de $\mathcal{S}$
$\mathcal{S}^n$	Un ensemble de n éléments de $\mathcal{S}$
$\mathcal{X}$	L'espace d'entrée
$\mathcal{Y}$	L'espace de sortie
$z = (x, y) \in \mathcal{X} \times \mathcal{Y}$	Un exemple étiqueté
$\mathbf{x}$	Un vecteur arbitraire
x_i	L'entrée d'indice i du vecteur $\mathbf{x}$
$\mathbf{M}$	Une matrice arbitraire
$\mathbf{I}$	La matrice identité
$M_{i,j}$	L'entrée de la ligne i et la colonne j de la matrice $\mathbf{M}$
$[\cdot]_+$	La fonction *hinge*
$\lvert\lvert \cdot \rvert\rvert$	Une norme arbitraire
$\lvert\lvert \cdot \rvert\rvert_p$	La norme L_p
A	Une série temporelle arbitraire
$x \sim P$	x est tiré de façon indépendante de la distribution de probabilités P
$\Pr[\cdot]$	La probabilité d'un événement
$\mathbb{E}[\cdot]$	L'espérance d'une variable aléatoire

Table C.1: Sommaire des notations.

Chapitre 3 Dans ce chapitre, nous analysons les méthodes existantes pour l'apprentissage de métriques supervisé et semi-supervisé. Nous commençons par présenter un aperçu de certaines métriques standard. Notre étude de l'état de l'art couvre les méthodes d'apprentissage de métriques supervisé dédiées aux vecteurs de *features* et aux séries temporelles. Après avoir discuté de ces aspects, nous pouvons conclure sur les avantages et les limites des méthodes récentes, qui représentent les principales motivations de nos contributions.

Chapitre 4 Dans ce chapitre, nous proposons un cadre générique innovant pour l'apprentissage d'une similarité en même temps que l'apprentissage d'un classifieur linéaire global à partir de vecteurs numériques. La formulation est capable d'exploiter les informations provenant de données non supervisées en plus d'un ensemble d'apprentissage étiqueté, ce qui en fait un cadre semi-supervisé. Nous montrons que notre formulation peut être utilisée avec une grande classe de régulariseurs et de nombreuses fonctions de similarité. Nous proposons une analyse théorique de ce cadre d'apprentissage conjoint à travers deux méthodes différentes, la robustesse algorithmique et la convergence uniforme basée sur la complexité de Rademacher, que nous comparons. Les résultats théoriques dérivés s'appliquent à la métrique apprise et au classifieur. De plus, ils sont génériques et couvrent des fonctions de similarité et des régulariseurs différents sans imposer de fortes contraintes. Les expériences menées sur des ensembles de données standard montrent les avantages de notre approche par rapport aux méthodes de pointe: JSL est efficace et performant.

Chapter 5 Le déformation temporelle dynamique (DTW, pour *Dynamic Time Warping* en anglais) est l'algorithme le plus connu pour mesurer la similarité entre deux séries temporelles en passant par l'alignement optimal entre elles. Malheureusement, comme nous l'avons vu dans le Chapitre 3, peu d'efforts de recherche ont été faits pour l'adapter à des séries temporelles multivariées et encore moins pour l'améliorer en apprenant. Dans ce chapitre, nous proposons une nouvelle méthode pour l'apprentissage de similarités basées sur DTW, afin d'améliorer la classification des séries temporelles. Dans cette contribution, nous utilisons également le cadre des similarités (ϵ, γ, τ)-bonnes (Balcan et al., 2008b), fournissant des garanties sur la performance d'un classifieur linéaire construit à partir de la métrique apprise. Montrer que notre méthode a une stabilité uniforme nous permet de dériver la première borne de généralisation d'une métrique apprise à partir de données temporelles. Nous effectuons des expériences sur des jeux de données temporelles multivariées des applications réelles, en comparant notre méthode aux approches de l'état de l'art. L'étude expérimentale montre que la méthode proposée est efficace, tout en fournissant des classifieurs parcimonieux.

Conclusion

Dans cette thèse, nous nous sommes intéréssés à l'état du développement de l'apprentissage de métriques pour certains types de problèmes, tout en présentant leurs principales limitations. Nous avons abordé un certain nombre de ces limitations dans nos contributions. Tout d'abord, sur les vecteurs de *features* dans un cadre général, puis nous avons exploré ces problématiques sur les données structurées avec une dimension temporelle. Un accent important a été mis sur les fondements théoriques qui guident les propositions, car nous estimons que les garanties de performance sont essentielles. Lors de l'élaboration de nos méthodes, nous avons également essayé de garder à l'esprit les aspects pratiques qui les rendraient utilisables.

Dans l'ensemble, nous avons abordé les limitations suivantes :

- Le manque de résultats théoriques pour l'apprentissage de métriques : bien que ces dernières années un plus grand nombre d'études aient porté sur l'élaboration d'analyses théoriques, deux tendances peuvent être observées : (i) certaines de ces études ne sont que théoriques et (ii) dans l'ensemble, seul un petit nombre des approches pratiques du domaine ont abordé cette question.
- Les contraintes artificielles imposées à la métrique apprise: la plupart des méthodes imposent les propriétés de PSD et de symétrie souvent parce qu'ils apprennent une distance de Mahalanobis. Au lieu de cela, nous avons choisi de travailler avec des fonctions de similarité qui n'ont pas besoin de ces propriétés.
- Le manque de résultats et de méthodes dans l'apprentissage de métriques pour les séries temporelles, en particulier pour le type le plus courant dans les applications réelles, les séries temporelles multivariées.

La première contribution a été de proposer JSL, un cadre qui permet d'apprendre une

fonction de similarité en même temps qu'un classifieur linéaire global. La méthode optimise directement une fonction de similarité (ϵ, γ, τ)-bonne (Balcan et al., 2008b), qui est directement liée à la performance du classifieur. Une caractéristique importante de JSL en découlant est son réglage semi-supervisé. Il permet à la méthode de tirer profit des données non étiquetées d'une manière différente de la plupart des méthodes semi-supervisées, c'est-à-dire en l'utilisant pour construire l'espace de représentation dans lequel les données sont projetées. La capacité à exploiter des exemples non-étiquetés rend le cadre proposé particulièrement adapté au contexte réel où l'annotation des données est coûteuse, et seul un petit nombre d'instances étiquetées est disponible. À notre connaissance, cette méthode est la première à apprendre la métrique et le classifieur en même temps de façon semi-supervisée. La généralité de JSL réside dans le fait qu'il peut être instancié avec une large gamme de fonctions de similarité et de régulariseurs, fournissant ainsi un moyen de résoudre des problèmes avec des propriétés différentes. La régularisation générale a la capacité d'incorporer les connaissances antérieures sur la tâche donnée. Nous proposons également une méthode efficace pour résoudre le problème, qui est convexe sous contraintes légères pour la fonction de similarité. D'un point de vue théorique, nous analysons JSL à travers la perspective de deux cadres: la robustesse algorithmique et la convergence uniforme avec la complexité de Rademacher. Les deux analyses donnent des bornes de généralisation pour la similarité et le classifieur appris qui sont valides pour différentes fonctions de similarité et fournissent des informations à leur sujet. La vaste étude expérimentale montre que JSL obtient de meilleures performances qu'un grand nombre d'autres méthodes. Le code source de JSL est disponible publiquement sous licence GNU/GPL 2+[1].

Notre deuxième contribution se concentre sur un type particulier de données, qui sont les séries temporelles multivariées. Nous avons proposé d'apprendre une fonction de similarité bilinéaire pour une tâche de classification. Une fois de plus, nous avons cherché à généraliser les noyaux en relâchant la contrainte PSD, tout en liant les propriétés de la similarité à sa performance en classification. Cela a été fait en apprenant une similarité (ϵ, γ, τ)-bonne basée sur l'alignement optimal entre les séries temporelles calculé par DTW. Le problème, appelé SLTS, est formulé comme un programme quadratique, qui peut être résolu efficacement. Travailler avec des séries temporelles est souvent coûteux en termes de calcul, principalement en raison de la complexité des données et de la DTW lors du calcul du meilleur alignement entre les paires de points. Nous avons pu réduire le nombre de valeurs de similarités et d'alignements à calculer en basant l'approche sur un ensemble de *landmarks* et limitant leur nombre. Nous avons établi des garanties de généralisation pour SLTS basées sur la stabilité uniforme, assurant la consistence de la métrique obtenue. La borne de généralisation dérivée est plutôt serrée et indépendante des longueurs des séries temporelles et des alignements, ressemblant à des bornes similaires pour les vecteurs numériques. L'étude expérimentale montre les bonnes propriétés de la métrique, ainsi que sa performance par rapport aux méthodes standard. À notre connaissance, SLTS est la première méthode d'apprentissage de métriques pour les séries temporelles multivariées qui est accompagnée d'une telle analyse théorique. Le code source pour la résolution de

[1] Télécharger à l'adresse `http://inicolae.com/resources/jsl.zip`.

SLTS est également disponible publiquement sous licence GNU/GPL 2+[2].

Il existe plusieurs perspectives intéressantes pour étendre les méthodes que nous avons proposées. L'une d'entre elle correspond à l'adaptation de l'apprentissage pour plusieurs métriques, chacune faisant référence à une région différente de l'espace. Nous pensons que cette direction peut capturer des informations supplémentaires dans les données, ainsi que fournir une manière implicite d'introduire la non-linéarité. Notez que le cadre des similarités (ϵ, γ, τ)-bonnes comprend une version dans laquelle plusieurs fonctions peuvent être combinées pour obtenir une métrique globale qui bénéficie des mêmes garanties. Une autre extension possible est d'adapter les approches proposées à un apprentissage incrémental, où les exemples arrivent un à la fois. En pratique, les premiers points reçus seraient considérés comme des *landmarks* pour le processus d'apprentissage. Pour ce cadre, on pourrait dériver des bornes de regret par rapport à la version ayant à disposition l'intégralité des données.

D'un point de vue théorique, cette thèse a analysé les garanties qui peuvent être obtenues par une fonction de similarité dans la classification linéaire. Ces résultats ont été obtenus à partir de la théorie des similarités (ϵ, γ, τ)-bonnes. Malheureusement, ce cadre lie les fonctions de similarité au cas particulier des classifieurs linéaires globaux. Considérant que l'état de l'art en apprentissage de métriques, qu'il s'agisse de vecteurs de *features* ou de séries temporelles, est orienté vers le classifieur de type k-NN, il serait important de développer des cadres théoriques d'analyse ou des résultats particuliers pour certaines méthodes concernant cette règle de classification locale. Certains résultats ont déjà été établis pour l'apprentissage incrémental et non-incrémental dans Qamar et al. (2008) sur la base de la théorie du perceptron de Freund & Schapire (1998).

Une autre voie prometteuse pour la recherche serait d'explorer des milieux qui n'ont pas reçu beaucoup d'attention dans l'apprentissage de métriques, comme l'apprentissage non-supervisé. Celui-ci un problème difficile, car le critère que la métrique doit optimiser n'est pas toujours évident, de même que les mesures de performance. Un premier résultat utilisant la stabilité uniforme pour dériver des garanties théoriques pour le cas général des algorithmes d'apprentissage non-supervisés a récemment été dérivé dans Abou-Moustafa & Schuurmans (2015), ouvrant la voie à d'autres recherches. Plus précisément, ils se concentrent sur les applications de *clustering* et de réduction de la dimensionnalité. Malheureusement, leurs résultats dépendent de la définition manuelle d'un critère d'évaluation pour chaque application.

[2]Télécharger à l'adresse `http://inicolae.com/resources/slts.zip`.

Bibliography

John Aach and George M. Church. Aligning Gene Expression Time Series with Time Warping Algorithms. *Bioinformatics*, 17(6):495–508, 2001.

Karim T. Abou-Moustafa and Dale Schuurmans. Generalization in Unsupervised Learning. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, 2015.

Ghazi Al-Naymat, Sanjay Chawla, and Javid Taheri. SparseDTW: A Novel Approach to Speed Up Dynamic Time Warping. In *Proceedings of the 8th Australasian Data Mining Conference (AusDM)*, pages 117–127. Australian Computer Society, Inc., 2009.

MOSEK ApS. *The MOSEK Python optimizer API manual. Version 7.0 (Revision 114)*, 2015.

Mahdieh S. Baghshah and Saeed B. Shouraki. Semi-Supervised Metric Learning Using Pairwise Constraints. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1217–1222. Morgan Kaufmann Publishers Inc., 2009.

Claus Bahlmann, Bernard Haasdonk, and Hans Burkhardt. On-Line Handwriting Recognition with Support Vector Machines: A Kernel Approach. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 49–54. IEEE Computer Society, 2002.

Maria-Florina Balcan and Avrim Blum. On a Theory of Learning with Similarity Functions. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 73–80. ACM, 2006.

Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. A Theory of Learning with Similarity Functions. *Machine Learning Journal (MLJ)*, 72(1):89–112, 2008a.

Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. Improved Guarantees for Learning via Similarity Functions. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT)*, pages 287–298. Omnipress, 2008b.

Jun-Peng Bao, Jun-Yi Shen, Xiao-Domg Liu, and Hai-Yan Liu. Quick Asymmetric Text Similarity Measures. *International Conference on Machine Learning and Cybernetics (ICMLC)*, 2003.

Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research (JMLR)*, 3: 463–482, 2003.

Peter L. Bartlett, Stéphane Boucheron, and Gábor Lugosi. Model Selection and Error Estimation. *Machine Learning Journal (MLJ)*, 48(1):85–113, 2002.

Gustavo E. A. P .A. Batista, Eamonn J. Keogh, Oben M. Tataw, and Vinícius M.A. de Souza. CID: An Efficient Complexity-Invariant Distance for Time Series. *Data Mining and Knowledge Discovery (DMKD)*, 28(3):634–669, 2014.

Aurélien Bellet. *Supervised Metric Learning with Generalization Guarantees.* dissertation, 2012.

Aurélien Bellet and Amaury Habrard. Robustness and Generalization for Metric Learning. *Neurocomputing*, 151(1):259–267, 2015.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. Learning Good Edit Similarities with Generalization Guarantees. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 188–203, 2011.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. Similarity Learning for Provably Accurate Sparse Linear Classification. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1871–1878, 2012.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. A Survey on Metric Learning for Feature Vectors and Structured Data. Technical report, 2013.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. *Metric Learning.* Morgan & Claypool Publishers, 2015.

Shai Ben-David, Nadav Eiron, and Philip M. Long. On the Difficulty of Approximately Maximizing Agreements. *Journal of Computer and System Sciences (CSS)*, 66(3): 496–514, 2003.

James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research (JMLR)*, 13:281–305, 2012.

Wei Bian and Dacheng Tao. Learning a Distance Metric by Empirical Loss Minimization. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1186–1191. IJCAI/AAAI, 2011.

Wei Bian and Dacheng Tao. Constrained Empirical Risk Minimization Framework for Distance Metric Learning. *IEEE Transactions on Neural Networks Learning Systems (TNNLS)*, 23(8):1194–1205, 2012.

Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating Constraints and Metric Learning in Semi-supervised Clustering. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 81–88. ACM, 2004.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., 2006.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th Workshop on Computational Learning Theory*, pages 144–152. ACM, 1992.

Stéphane Boucheron, Olivier Bousquet, and Gábor Lugosi. Theory of Classification: A Survey of Some Recent Advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

Olivier Bousquet and André Elisseeff. Algorithmic Stability and Generalization Performance. In *Advances in Neural Information Processing Systems (NIPS)*, pages 196–202. Max-Planck-Gesellschaft, MIT Press, 2001.

Olivier Bousquet and André Elisseeff. Stability and Generalization. *Journal of Machine Learning Research (JMLR)*, 2:499–526, 2002.

Qiong Cao, Zheng-Chu Guo, and Yiming Ying. Generalization Bounds for Metric and Similarity Learning. *ArXiv e-prints*, arXiv:1207.5437, 2012.

Olivier Chapelle, Bernard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. An Online Algorithm for Large Scale Image Similarity Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 306–314. Curran Associates, Inc., 2009.

Huanhuan Chen, Fengzhen Tang, Peter Tiňo, Anthony G. Cohn, and Xin Yao. Model Metric Co-Learning for Time Series Classification. In *Proceedings of the 26nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3387–3394. AAAI Press, 2015.

Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and Fast Similarity Search for Moving Object Trajectories. In *Proceedings of ACM SIGMOD International Conference on Management of Data (ICMD)*, pages 491–502. ACM, 2005.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 539–546. IEEE Computer Society, 2005.

Gabriella Contardo, Ludovic Denoyer, and Thierry Artières. Recurrent Neural Networks for Adaptive Feature Acquisition. In *Proceedings of the 23nd International Conference on Neural Information Processing (ICONIP)*, pages 591–599, 2016.

Corinna Cortes and Vladimir N. Vapnik. Support-Vector Networks. *Machine Learning Journal (MLJ)*, 20(3):273–297, 1995a.

Corinna Cortes and Vladimir N. Vapnik. Support-Vector Networks. *Machine Learning Journal (MLJ)*, 20(3):273–297, 1995b.

Thomas M. Cover and P. E. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory (TIT)*, 13(1):21–27, 1967.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research (JMLR)*, 7:551–585, 2006.

Marco Cuturi. Fast Global Alignment Kernels. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 929–936, 2011.

Marco Cuturi, Jean-Philippe Vert, Oystein Birkenes, and Tomoko Matsui. A Kernel for Time Series Based on Global Alignments. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 413–416, 2007.

Asma Dachraoui, Alexis Bondu, and Antoine Cornuéjols. Early Classification of Time Series as a Non Myopic Sequential Decision Making Problem. pages 433–447, 2015.

Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-Theoretic Metric Learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 209–216. ACM, 2007.

Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

Sahibsingh A. Dudani. The Distance-Weighted k-Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man, and Cybernetics (TSMC)*, SMC-6(4):325–327, 1976.

Information Retrieval: Data Structures and Algorithms. Prentice-Hall, Inc., 1992.

Cédric Frambourg, Ahlame Douzal-Chouakria, and Éric Gaussier. Learning Multiple Temporal Matching for Time Series Classification. In *International Symposium on Intelligent Data Analysis (IDA)*, volume 8207 of *Lecture Notes in Computer Science*, pages 198–209. Springer, 2013.

Yoav Freund and Robert E. Schapire. A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. In *Proceedings of the 2nd European Conference on Computational Learning Theory (EuroCOLT)*, pages 23–37. Springer-Verlag, 1995.

Yoav Freund and Robert E. Schapire. Large Margin Classification Using the Perceptron Algorithm. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*, pages 209–217. ACM, 1998.

Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29:1189–1232, 2000.

Geoffrey W. Gates. The Reduced Nearest Neighbor Rule. *IEEE Transactions on Information Theory (TIT)*, 18:431–433, 1972.

Evarist Gine and Joel Zinn. Some Limit Theorems for Empirical Processes. *The Annals of Probability*, 12(4):929–989, 1984.

Amir Globerson and Sam Roweis. Metric Learning by Collapsing Classes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 18, pages 451–458. MIT Press, 2006.

Jacob Goldberger, Sam Roweis, Geoffrey Hinton, and Ruslan Salakhutdinov. Neighbourhood Components Analysis. In *Advances in Neural Information Processing Systems (NIPS)*, pages 513–520. MIT Press, 2004.

Michał Grabowski and Andrzej Szałas. A Technique for Learning Similarities on Complex Structures with Applications to Extracting Ontologies. In *Proceedings of the 3rd Atlantic Web Intelligence Conference*. Springer Verlag, 2000.

Zheng-Chu Guo and Yiming Ying. Guaranteed Classification via Regularized Similarity Learning. *ArXiv e-prints*, arXiv:1306.3108, 2013.

Zheng-Chu Guo, Yiming Ying, and Ding-Xuan Zhou. Online Regularized Learning with Pairwise Loss Functions. *Advances in Computational Mathematics (ACM)*, pages 1–24, 2016.

Isabelle Guyon. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research (JMLR)*, 3:1157–1182, 2003.

Richard W. Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 26(2):147–160, 1950.

P. E. Hart. The Condensed Nearest Neighbor Rule. *IEEE Transactions on Information Theory (TIT)*, 14:515–516, 1968.

Daniel S. Hirschberg. A Linear Space Algorithm for Computing Maximal Common Subsequences. *Communications of the ACM*, 18(6):341–343, 1975.

Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association (JASA)*, 58(301):13–30, 1963.

Steven C.-H. Hoi, Wei Liu, and Shih-Fu Chang. Semi-Supervised Distance Metric Learning for Collaborative Image Retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

Steven C.-H. Hoi, Wei Liu, and Shih-Fu Chang. Semi-Supervised Distance Metric Learning for Collaborative Image Retrieval and Clustering. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 6(3):1–26, 2010.

Zhouyuan Huo, Feiping Nie, and Heng Huang. Robust and Effective Metric Learning Using Capped Trace Norm: Metric Learning via Capped Trace Norm. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1605–1614. ACM, 2016.

Armin Hust. Learning Similarities for Collaborative Information Retrieval. In *Proceedings of KI Workshop Machine Learning and Interaction for Text-Based Information Retrieval, TIR*, pages 43–54, 2004.

Yoshiharu Ishikawa, Ravishankar Subramanya, and Christos Faloutsos. Mindreader: Querying Databases Through Multiple Examples. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB)*, pages 218–227. Morgan Kaufmann Publishers Inc., 1998.

Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing (TASSP)*, 23(1):67–72, 1975.

Edwin T. Jaynes. Information Theory and Statistical Mechanics. *Physical Review*, 106: 620–630, 1957.

Rong Jin, Shijun Wang, and Yang Zhou. Regularized Distance Metric Learning: Theory and Algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 862–870. Curran Associates, Inc., 2009.

Ian T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

Purushottam Kar and Prateek Jain. Similarity-Based Learning via Data Driven Embeddings. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1998–2006. Curran Associates, Inc., 2011.

Leonard Kaufman and Peter J. Rousseeuw. Clustering by Means of Medoids. In *Statistical Data Analysis Based on the L1-Norm and Related Methods*, pages 405–416. North-Holland, 1987.

Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley series in probability and mathematical statistics. Wiley, 1990.

Michael J. Kearns and Dana Ron. Algorithmic Stability and Sanity-Check Bounds for Leave-One-Out Cross-Validation. In *Proceedings of the 10th Annual Conference on Computational Learning Theory (COLT)*, pages 152–162, 1997.

Michael J. Kearns and Robert E. Schapire. Efficient Distribution-Free Learning of Probabilistic Concepts. *Journal of Computer and System Sciences (JCSS)*, 48(3):464–497, 1994.

Dor Kedem, Stephen Tyree, Kilian Q. Weinberger, Fei Sha, and Gert Lanckriet. Non-Linear Metric Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2582–2590. 2012.

Eamonn J. Keogh and Chotirat A. Ratanamahatana. Everything You Know About Dynamic Time Warping is Wrong. *3rd Workshop on Mining Temporal and Sequential Data*, 2004.

David Knoke and Peter J. Burke. *Log-Linear Models*. Number v. 20; v. 1980 in A Sage university paper. SAGE Publications, 1980.

Andrey Kolmogorov and Viktor Tikhomirov. ϵ-Entropy and ϵ-Capacity of Sets in Functional Spaces. *American Mathematical Society Translations (AMST)*, 2(17):277–364, 1961.

Vladimir Koltchinskii. On the Central Limit Theorem for Empirical Measures. *Theory Probability and Mathematical Statistics*, 24:63–75, 1981.

Vladimir Koltchinskii. Rademacher Penalties and Structural Risk Minimization. *IEEE Transactions on Information Theory (TIT)*, 47(5):1902–1914, 2001.

Joseph Z. B. Kruskall and Mark Liberman. The Symmetric Time-Warping Problem: From Continuous to Discrete. In *Time Warps, String Edits and Macromolecules: The Theory and Practice of String Comparison*. Addison-Wesley, 1983.

Brian Kulis. Metric Learning: A Survey. *Foundations and Trends in Machine Learning*, 5 (4):287–364, 2013.

Mahesh Kumar, Nitin R. Patel, and Jonathan Woo. Clustering Seasonality Patterns in the Presence of Errors. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 557–563. ACM, 2002.

Rémi Lajugie, Damien Garreau, Francis Bach, and Sylvain Arlot. Metric Learning for Temporal Sequence Alignment. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1817–1825. Curran Associates, Inc., 2014.

Marc T. Law, Nicolas Thome, and Matthieu Cord. Quadruplet-Wise Image Similarity Learning. In *IEEE International Conference on Computer Vision (ICCV)*, pages 249–256, 2013.

Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer, 1991.

Vladimir Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics-Doklady*, 10(8):707–710, 1966.

Moshe Lichman. UCI Machine Learning Repository, 2013.

Stuart P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory (TIT)*, 28(2):129–137, 2006.

Prasanta C. Mahalanobis. On the Generalised Distance in Statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, 1936.

David Maier. The Complexity of Some Problems on Subsequences and Supersequences. *ACM Journal*, 25(2):322–336, 1978.

Colin McDiarmid. On the Method of Bounded Differences. In *Surveys in Combinatorics*, pages 148–188. Cambridge University Press, 1989.

Jiangyuan Mei, Meizhu Liu, Yuan-Fang Wang, and Huijun Gao. Learning a Mahalanobis Distance-Based Dynamic Time Warping Measure for Multivariate Time Series Classification. *IEEE Transactions on Cybernetics (TC)*, 2015.

James Mercer. Functions of Positive and Negative Type, and Their Connection with the Theory of Integral Equations. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 209(441-458):415–446, 1909.

Carla S. Möller-Levet, Frank Klawonn, Kwang-Hyun Cho, and Olaf Wolkenhauer. *Fuzzy Clustering of Short Time-Series and Unevenly Distributed Sampling Points*, pages 330–340. Springer Berlin Heidelberg, 2003.

Maria-Irina Nicolae, Marc Sebban, Amaury Habrard, Éric Gaussier, and Massih-Reza Amini. Algorithmic Robustness for Learning via (ϵ, γ, τ)-Good Similarity Functions. *ArXiv e-prints*, arxiv:1412.6452, 2014.

Maria-Irina Nicolae, Éric Gaussier, Amaury Habrard, and Marc Sebban. Apprentissage joint semi-supervisé de bonnes similarités. In *French Conference on Machine Learning (CAp)*, 2015a.

Maria-Irina Nicolae, Éric Gaussier, Amaury Habrard, and Marc Sebban. Joint semi-supervised similarity learning for linear classification. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 594–609, 2015b.

Maria-Irina Nicolae, Marc Sebban, Amaury Habrard, Éric Gaussier, and Massih-Reza Amini. Algorithmic robustness for semi-supervised (ϵ, γ, τ)-good metric learning. In *Proceedings of the 22nd International Conference on Neural Information Processing (ICONIP)*, pages 253–263, 2015c.

Maria-Irina Nicolae, Éric Gaussier, Amaury Habrard, and Marc Sebban. Apprentissage de similarités pour la classification de séries temporelles multivariées. In *French Conference on Machine Learning (CAp)*, 2016.

Gang Niu, Bo Dai, Makoto Yamada, and Masashi Sugiyama. Information-Theoretic Semi-Supervised Metric Learning via Entropy Regularization. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*. icml.cc / Omnipress, 2012.

Sinno J. Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(10):1345–1359, 2010.

Spiros Papadimitriou, Jimeng Sun, and Christos Faloutsos. Streaming Pattern Discovery in Multiple Time-Series. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 697–708. VLDB Endowment, 2005.

Michaël Perrot and Amaury Habrard. Regressive Virtual Metric Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1801–1809, 2015.

David Pollard. A Central Limit Theorem for k-Means Clustering. *The Annals of Probability*, 10(4):919–926, 1982.

Frits H. Post, Gregory M. Nielson, and Georges-Pierre Bonneau. *Data Visualization: The State of the Art*. The Springer International Series in Engineering and Computer Science. Springer US, 2002.

Zoltán Prekopcsák and Daniel Lemire. Time Series Classification by Class-Specific Mahalanobis Distance Measures. *Advances in Data Analysis and Classification (ADAC)*, 6(3): 185–200, 2012.

Ali M. Qamar and Éric Gaussier. Online and Batch Learning of Generalized Cosine Similarities. In *IEEE International Conference on Data Mining (ICDM)*, pages 926–931, 2009a.

Ali M. Qamar and Éric Gaussier. Similarity Learning in Nearest Neighbor and Application to Information Retrieval. In *Proceedings of the 3rd BCS-IRSG Conference on Future Directions in Information Access (FDIA)*, pages 131–133. British Computer Society, 2009b.

Ali M. Qamar, Éric Gaussier, Jean-Pierre Chevallet, and Joo H. Lim. Similarity Learning for Nearest Neighbor Classification. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 983–988, 2008.

Chotirat A. Ratanamahatana and Eamonn J. Keogh. Making Time-Series Classification More Accurate Using Learned Constraints. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 11–22, 2004.

Hiroaki Sakoe and Seibi Chiba. Dynamic-Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics Speech and Signal Processing (TASSP)*, 26(1):43–49, 1978.

Stan Salvador and Philip Chan. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. In *KDD Workshop on Mining Temporal and Sequential Data*, 2004.

Bernhard Schölkopf and Alexander J. Smola. Learning with Kernels, 2002.

Matthew Schultz and Thorsten Joachims. Learning a Distance Metric from Relative Comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, pages 41–48, 2003.

Shai Shalev-Shwartz, Yoram Singer, and Andrew Y. Ng. Online and Batch Learning of Pseudo-Metrics. In *Proceedings of the 21st International Conference on Machine Learning (ICML*. ACM, 2004.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

Jingyi Shen, Weiping Huang, Dongyang Zhu, and Jun Liang. A Novel Similarity Measure Model for Multivariate Time Series Based on LMNN and DTW. *Neural Processing Letters (NPL)*, pages 1–13, 2016.

Yuan Shi, Aurélien Bellet, and Fei Sha. Sparse Compositional Metric Learning. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 2078–2084, 2014.

Hiroshi Shimodaira, Ken-Ichi Noma, Mitsuru Nakai, and Shigeki Sagayama. Dynamic Time-Alignment Kernel in Support Vector Machine. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2002.

Mohammad Shokoohi-Yekta, Jung Wang, and Eamonn Keogh. On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case. In *Proceedings of the SIAM International Conference on Data Mining (ICDM)*, pages 39–48, 2015.

Bernaard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1986.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large Scale Multiple Kernel Learning. *Journal of Machine Learning Research (JMLR)*, 7:1531–1565, 2006.

Jimeng Sun, Daby Sow, Jianying Hu, and Shahram Ebadollahi. Localized Supervised Metric Learning on Temporal Physiological Data. In *Proceedings of the 20th International Conference on Pattern Recognition (ICPR)*, pages 4149–4152, 2010.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1st edition, 1998.

Gineke A. ten Holt, Marcel J. T. Reinders, and Emile A. Hendriks. Multi-dimensional dynamic time warping for gesture recognition. In *13th Conference of the Advanced School for Computing and Imaging*, 2007.

Paolo Tormene, Toni Giorgino, Silvana Quaglini, and Mario Stefanelli. Matching Incomplete Time Series with Dynamic Time Warping: An Algorithm and an Application to Post-Stroke Rehabilitation. *Artificial Intelligence in Medicine*, 45(1):11–34, 2009.

Leslie G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11): 1134–1142, 1984.

Aad W. van der Vaart and Jon A. Wellner. *Weak Convergence and Empirical Processes*. Springer series in statistics. Springer, 1996.

Vladimi N. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, 1979.

Vladimir N Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Vladimir N. Vapnik and Alexey Y. Chervonenkis. A Note on One Class of Perceptrons. *Automation and Remote Control*, 25, 1964.

Vladimir N. Vapnik and Alexey Y. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability and Its Applications (TPA)*, 16(2):264–280, 1971.

Nakul Verma and Kristin Branson. Sample Complexity of Learning Mahalanobis Distance Metrics. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2584–2592, 2015.

Cédric Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer, 2009.

Vo T. Vinh and Duong T. Anh. Compression Rate Distance Measure for Time Series. In *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2015.

Michail Vlachos, Dimitrios Gunopulos, and Gautam Das. *Indexing Time Series Under Conditions of Noise*, pages 67–100. World Scientific Publishing, 2004.

Fei Wang and Changshui Zhang. Feature Extraction by Maximizing the Average Neighborhood Margin. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

Jun Wang, Alexandros Kalousis, and Adam Woznica. Parametric Local Metric Learning for Nearest Neighbor Classification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1601–1609. Curran Associates, Inc., 2012a.

Jun Wang, Adam Woznica, and Alexandros Kalousis. Learning Neighborhoods for Metric Learning. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 223–236. Springer Berlin Heidelberg, 2012b.

T. Warren Liao. Clustering of Time Series Data – A Survey. *Pattern Recognition Letters (PRL)*, 38(11):1857–1874, 2005.

Kilian Q. Weinberger and Lawrence K. Saul. Fast Solvers and Efficient Implementations for Distance Metric Learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1160–1167. ACM, 2008.

Kilian Q. Weinberger and Lawrence K. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *The Journal of Machine Learning Research (JMLR)*, 10:207–244, 2009.

Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2006.

Xiaoqing Weng and Junyi Shen. Classification of Multivariate Time Series Using Locality Preserving Projections. *Knowledge-Based Systems*, 21(7):581–587, 2008.

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance Metric Learning, with Application to Clustering with Side-Information. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 505–512, 2002.

Huan Xu and Shie Mannor. Robustness and Generalization. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*, pages 503–515, 2010.

Huan Xu and Shie Mannor. Robustness and Generalization. *Machine Learning Journal (MLJ)*, 86(3):391–423, 2012.

Liu Yang. Distance Metric Learning: A Comprehensive Survey, 2006.

Dit-Yan Yeung and Hong Chang. A Kernel Approach for Semisupervised Metric Learning. *IEEE Transactions on Neural Networks (TNN)*, 18:141–149, 2007.

Xuesong Yin, Songcan Chen, Enliang Hu, and Daoqiang Zhang. Semi-Supervised Clustering with Metric Learning: an Adaptive Kernel Method. *Pattern Recognition Letters (PRL)*, 43(4):1320–1333, 2010.

Daren Yu, Xiao Yu, Qinghua Hu, Jinfu Liu, and Anqi Wu. Dynamic Time Warping Constraints Learning for Large Margin Nearest Neighbor Classification. *Information Science*, 181:2787–2796, 2011.

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. Metric Learning as Convex Combinations of Local Models with Generalization Guarantees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Zheng-Jun Zha, Tao Mei, Meng Wang, Zengfu Wang, and Xian-Sheng Hua. Robust Distance Metric Learning with Auxiliary Knowledge. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1327–1332. Morgan Kaufmann Publishers Inc., 2009.

Jiaping Zhao, Zerong Xi, and Laurent Itti. MetricDTW: Local Distance Metric Learning in Dynamic Time Warping. *ArXiv e-prints*, arXiv:1606.03628, 2016.

Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-Norm Support Vector Machines. *Advances in Neural Information Processing Systems (NIPS)*, 16(1):49–56, 2004.

Xiaojin Zhu. *Semi-Supervised Learning with Graphs*. PhD thesis, 2005.

Abstract

The notion of metric plays a key role in machine learning problems, such as classification, clustering and ranking. Learning metrics from training data in order to make them adapted to the task at hand has attracted a growing interest in the past years. This research field, known as metric learning, usually aims at finding the best parameters for a given metric under some constraints from the data. The learned metric is used in a machine learning algorithm in hopes of improving performance. Most of the metric learning algorithms focus on learning the parameters of Mahalanobis distances for feature vectors. Current state of the art methods scale well for datasets of significant size. On the other hand, the more complex topic of multivariate time series has received only limited attention, despite the omnipresence of this type of data in applications. An important part of the research on time series is based on the dynamic time warping (DTW) computing the optimal alignment between two time series. The current state of metric learning suffers from some significant limitations which we aim to address in this thesis. The most important one is probably the lack of theoretical guarantees for the learned metric and its performance for classification. The theory of (ϵ, γ, τ)-good similarity functions has been one of the first results relating the properties of a similarity to its classification performance. A second limitation in metric learning comes from the fact that most methods work with metrics that enforce distance properties, which are computationally expensive and often not justified. In this thesis, we address these limitations through two main contributions. The first one is a novel general framework for jointly learning a similarity function and a linear classifier. This formulation is inspired from the (ϵ, γ, τ)-good theory, providing a link between the similarity and the linear classifier. It is also convex for a broad range of similarity functions and regularizers. We derive two equivalent generalization bounds through the frameworks of algorithmic robustness and uniform convergence using the Rademacher complexity, proving the good theoretical properties of our framework. Our second contribution is a method for learning similarity functions based on DTW for multivariate time series classification. The formulation is convex and makes use of the (ϵ, γ, τ)-good framework for relating the performance of the metric to that of its associated linear classifier. Using uniform stability arguments, we prove the consistency of the learned similarity leading to the derivation of a generalization bound.

Résumé

La notion de métrique joue un rôle clef dans les problèmes d'apprentissage automatique tels que la classification, le *clustering* et le *ranking*. L'apprentissage à partir de données de métriques adaptées à une tâche spécifique a suscité un intérêt croissant ces dernières années. Ce domaine vise généralement à trouver les meilleurs paramètres pour une métrique donnée sous certaines contraintes imposées par les données. La métrique apprise est utilisée dans un algorithme d'apprentissage automatique dans le but d'améliorer sa performance. La plupart des méthodes d'apprentissage de métriques optimisent les paramètres d'une distance de Mahalanobis pour des vecteurs de *features*. Les méthodes actuelles de l'état de l'art arrivent à traiter des jeux de données de tailles significatives. En revanche, le sujet plus complexe des séries temporelles multivariées n'a reçu qu'une attention limitée, malgré l'omniprésence de ce type de données dans les applications réelles. Une importante partie de la recherche sur les séries temporelles est basée sur la *dynamic time warping* (DTW), qui détermine l'alignement optimal entre deux séries temporelles. L'état actuel de l'apprentissage de métriques souffre de certaines limitations. La plus importante est probablement le manque de garanties théoriques concernant la métrique apprise et sa performance pour la classification. La théorie des fonctions de similarité (ϵ, γ, τ)-bonnes a été l'un des premiers résultats liant les propriétés d'une similarité à celles du classifieur qui l'utilise. Une deuxième limitation vient du fait que la plupart des méthodes imposent des propriétés de distance, qui sont coûteuses en terme de calcul et souvent non justifiées. Dans cette thèse, nous abordons les limitations précédentes à travers deux contributions principales. La première est un nouveau cadre général pour l'apprentissage conjoint d'une fonction de similarité et d'un classifieur linéaire. Cette formulation est inspirée de la théorie de similarités (ϵ, γ, τ)-bonnes, fournissant un lien entre la similarité et le classifieur linéaire. Elle est convexe pour une large gamme de fonctions de similarité et de régulariseurs. Nous dérivons deux bornes de généralisation équivalentes à travers les cadres de robustesse algorithmique et de convergence uniforme basée sur la complexité de Rademacher, prouvant les propriétés théoriques de notre formulation. Notre deuxième contribution est une méthode d'apprentissage de similarités basée sur DTW pour la classification de séries temporelles multivariées. Le problème est convexe et utilise la théorie des fonctions (ϵ, γ, τ)-bonnes liant la performance de la métrique à celle du classifieur linéaire associé. A l'aide de la stabilité uniforme, nous prouvons la consistance de la similarité apprise conduisant à la dérivation d'une borne de généralisation.