HAL open science

Supervised Metric Learning with Generalization Guarantees

Aurélien Bellet

To cite this version:

Aurélien Bellet. Supervised Metric Learning with Generalization Guarantees. Machine Learning [cs.LG]. Université Jean Monnet - Saint-Etienne, 2012. English. NNT : . tel-00770627v4

HAL Id: tel-00770627

https://theses.hal.science/tel-00770627v4

Submitted on 10 Jul 2013 (v4), last revised 25 Feb 2014 (v6)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

École Doctorale ED488 “Sciences, Ingénierie, Santé”

Supervised Metric Learning with Generalization Guarantees

Thèse avec label européen préparée par **Aurélien Bellet**
pour obtenir le grade de :

Docteur de l’Université Jean Monnet de Saint-Étienne
Domaine : **Informatique**

Laboratoire Hubert Curien, UMR CNRS 5516
Faculté des Sciences et Techniques

Soutenance le 11 Décembre 2012 au Laboratoire Hubert Curien
devant le jury composé de :

Pierre **Dupont**	Professeur, Université Catholique de Louvain	Rapporteur
Rémi **Gilleron**	Professeur, Université de Lille	Examinateur
Amaury **Habrard**	Professeur, Université de Saint-Étienne	Co-directeur
Jose **Oncina**	Professeur, Universidad de Alicante	Rapporteur
Liva **Ralaivola**	Professeur, Aix-Marseille Université	Examinateur
Marc **Sebban**	Professeur, Université de Saint-Étienne	Directeur

---

Remerciements

Je tiens tout d'abord à remercier Pierre Dupont, Professeur à l'Université Catholique de Louvain, et Jose Oncina, Professeur à l'Université d'Alicante, d'avoir accepté d'être les rapporteurs de mon travail de thèse. Leurs remarques pertinentes m'ont permis d'améliorer la qualité de ce manuscrit. Plus généralement, je remercie l'ensemble du jury, notamment Rémi Gilleron, Professeur à l'Université de Lille, et Liva Ralaivola, Professeur à Aix-Marseille Université, qui ont tout de suite accepté d'être examinateurs.

Je remercie chaleureusement mon directeur et mon co-directeur de thèse, Marc et Amaury, avec qui j'ai développé des liens professionnels et personnels qui de toute évidence dureront au-delà de cette thèse. Je suis particulièrement reconnaissant envers Marc qui, malgré son attrait pour une certaine équipe de football, a su me convaincre de faire cette thèse et m'a fait confiance en acceptant un arrangement extraordinaire (dans tous les sens du terme) pour que je puisse passer ma première année à Édimbourg.

Je veux également saluer les collègues du Laboratoire Hubert Curien et du département d'informatique de l'UJM. En premier lieu, mon voisin de bureau et ami JP, qui fut aussi un excellent co-équipier Warlight. Je pense aussi aux autres doctorants (anciens et actuels) que sont Laurent, Christophe, Émilie, David, Fabien, Tung, Chahrazed et Mattias. Enfin, je veux mentionner les personnes rencontrées dans le cadre des projets PASCAL2 et LAMPADA, notamment Emilie et Pierre du LIF de Marseille avec qui j'espère avoir l'occasion de travailler et de collaborer encore dans le futur.

D'un point de vue plus personnel, je salue évidemment les amis, qui sont trop nombreux pour être cités mais qui se reconnaîtront. La présence de certains à ma soutenance me fait énormément plaisir. Ces remerciements ne seraient pas complets sans un mot pour Marion, qui m'a beaucoup soutenu et encouragé pendant ces (presque) trois années. Elle a même essayé de s'intéresser à la classification linéaire parcimonieuse, réussissant à faire illusion lors d'une réception à ECML! Enfin, *last but not least*, je dédie tout simplement cette thèse à mes parents, mes grands-parents et mon petit frère.

Contents

List of Figures vii

List of Tables ix

1 Introduction 1

I Background 7

2 Preliminaries 9
- 2.1 Supervised Learning . . . 9
- 2.2 Deriving Generalization Guarantees . . . 14
- 2.3 Metrics . . . 19
- 2.4 Conclusion . . . 27

3 A Review of Supervised Metric Learning 29
- 3.1 Introduction . . . 29
- 3.2 Metric Learning from Feature Vectors . . . 30
- 3.3 Metric Learning from Structured data . . . 42
- 3.4 Conclusion . . . 45

II Contributions in Metric Learning from Structured Data 49

4 A String Kernel Based on Learned Edit Similarities 51
- 4.1 Introduction . . . 51
- 4.2 A New Marginalized String Edit Kernel . . . 52
- 4.3 Computing the Edit Kernel . . . 53
- 4.4 Experimental Validation . . . 59
- 4.5 Conclusion . . . 63

5 Learning Good Edit Similarities from Local Constraints 65
- 5.1 Introduction . . . 65
- 5.2 The Theory of (ϵ, γ, τ)-Good Similarity Functions . . . 67
- 5.3 Preliminary Experimental Study . . . 71
- 5.4 Learning (ϵ, γ, τ)-Good Edit Similarity Functions . . . 77
- 5.5 Theoretical Analysis . . . 81
- 5.6 Experimental Validation . . . 89
- 5.7 Conclusion . . . 95

III Contributions in Metric Learning from Feature Vectors **99**

6 Learning Good Bilinear Similarities from Global Constraints **101**
6.1 Introduction . . . 101
6.2 Learning (ϵ, γ, τ)-Good Bilinear Similarity Functions . . . 102
6.3 Theoretical Analysis . . . 105
6.4 Experimental Validation . . . 109
6.5 Conclusion . . . 113

7 Robustness and Generalization for Metric Learning **117**
7.1 Introduction . . . 117
7.2 Robustness and Generalization for Metric Learning . . . 118
7.3 Necessity of Robustness . . . 124
7.4 Examples of Robust Metric Learning Algorithms . . . 126
7.5 Conclusion . . . 129

8 Conclusion & Perspectives **131**

List of Publications **135**

A Learning Conditional Edit Probabilities **137**

B Proofs **141**
B.1 Proofs of Chapter 5 . . . 141
B.2 Proofs of Chapter 7 . . . 147

Bibliography **153**

List of Figures

1.1 The two-fold problem of generalization in metric learning 3

2.1 3D unit balls of the L_1, L_2 and $L_{2,1}$ norms 13
2.2 Geometric interpretation of L_2 and L_1 constraints 13
2.3 Plot of several loss functions for binary classification 14
2.4 Minkowski distances: unit circles for various values of p 22
2.5 Strategies to delete a node within a tree 25

3.1 Intuition behind metric learning 30

4.1 An example of memoryless cPFT 55
4.2 Two cPFT $T|\mathtt{a}$ and $T|\mathtt{ab}$ modeling $p_e(\mathtt{s}|\mathtt{a})$ and $p_e(\mathtt{s}|\mathtt{ab})$ 56
4.3 The cPFT $T|\mathtt{a}$ and $T|\mathtt{ab}$ represented in the form of automata 56
4.4 Automaton modeling the intersection of the automata of Figure 4.3 57
4.5 A handwritten digit and its string representation 60
4.6 Comparison of our edit kernel with edit distances 61
4.7 Comparison of our edit kernel with other string kernels 63
4.8 Influence of the parameter t of $K_{L\&J}$ 64

5.1 A graphical insight into (ϵ, γ, τ)-goodness 68
5.2 Projection space implied by the toy example of Figure 5.1 69
5.3 Estimation of the goodness of edit similarities 72
5.4 Classification accuracy and sparsity (Digit dataset) 74
5.5 Classification accuracy and sparsity with respect to λ 75
5.6 Classification accuracy and sparsity with respect to t 75
5.7 Classification accuracy and sparsity (Word dataset) 76
5.8 Learning the edit costs: rate of convergence (Word dataset) 90
5.9 Influence of the pairing strategies (Word dataset) 91
5.10 Learning the separator: accuracy and sparsity results (Word dataset) . . . 92
5.11 Learning the edit costs: rate of convergence (Digit dataset) 94
5.12 Influence of the pairing strategies (Digit dataset) 95
5.13 Example of a set of reasonable points (Digit dataset) 95
5.14 1-Nearest Neighbor results (Word dataset) 97

6.1 Accuracy of the methods with respect to KPCA dimension 112
6.2 Feature space induced by the similarity (Rings dataset) 113
6.3 Feature space induced by the similarity (Svmguide1 dataset) 114

7.1 Illustration of robustness in the classic and metric learning settings 120

List of Tables

1.1 Summary of notation 6

2.1 Common regularizers on vectors 12
2.2 Common regularizers on matrices 12
2.3 Example of an edit cost matrix 24

3.1 Metric learning from feature vectors: main features of the methods 46
3.2 Metric learning from structured data: main features of the methods . . . 46

4.1 Statistical comparison of our edit kernel with edit distances 61
4.2 Statistical comparison of our edit kernel with other string kernels 63

5.1 Example of a set of reasonable points (Word dataset) 92
5.2 Discriminative patterns extracted from the reasonable points of Table 5.1 93
5.3 Summary of the main features of GESL 96
5.4 1-Nearest Neighbor results on the Digit dataset 97

6.1 Properties of the datasets used in the experimental study 110
6.2 Accuracy of the linear classifiers built from the studied similarities 111
6.3 Accuracy of 3-NN classifiers using the studied similarities 111
6.4 Runtime of the studied metric learning methods 113
6.5 Summary of the main features of SLLC 114

"There is nothing more practical than a good theory."

— James C. Maxwell

"There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable.

There is another theory which states that this has already happened."

— Douglas Adams

CHAPTER 1
Introduction

The goal of machine learning is to automatically figure out how to perform tasks by generalizing from examples. A machine learning algorithm takes a data sample as input and infers a model that captures the underlying mechanism (usually assumed to be some unknown probability distribution) which generated the data. Data can consist of features vectors (e.g., the age, body mass index, blood pressure, ... of a patient) or can be structured, such as strings (e.g., text documents) or trees (e.g., XML documents). A classic setting is *supervised learning*, where the algorithm has access to a set of training examples along with their labels and must learn a model that is able to accurately predict the label of future (unseen) examples. Supervised learning encompasses classification problems, where the label set is finite (for instance, predicting the label of a character in a handwriting recognition system) and regression problems, where the label set is continuous (for example, the temperature in weather forecasting). On the other hand, an *unsupervised learning* algorithm has no access to the labels of the training data. A classic example is clustering, where we aim at assigning data into similar groups. The generalization ability of the learned model (i.e., its performance on unseen examples) can sometimes be guaranteed using arguments from statistical learning theory.

Relying on the saying "birds of a feather flock together", many supervised and unsupervised machine learning algorithms are based on a notion of *metric* (similarity or distance function) between examples, such as k-nearest neighbors or support vector machines in the supervised setting and K-Means clustering in unsupervised learning. The performance of these algorithms critically depends on the relevance of the metric to the problem at hand — for instance, we hope that it identifies as similar the examples that share the same underlying label and as dissimilar those of different labels. Unfortunately, standard metrics (such as the Euclidean distance between feature vectors or the edit distance between strings) are often not appropriate because they fail to capture the specific nature of the problem of interest.

For this reason, a lot of effort has gone into *metric learning*, the research topic devoted to automatically learning metrics from data. In this thesis, we focus on supervised metric learning, where we try to adapt the metric to the problem at hand using the information brought by a sample of labeled examples. Many of these methods aim to find the parameters of a metric so that it best satisfies a set of local constraints over the training

sample, requiring for instance that pairs of examples of the same class should be similar and that those of different class should be dissimilar according to the learned metric. A large body of work has been devoted to supervised metric learning from feature vectors, in particular Mahalanobis distance learning, which essentially learns a linear projection of the data into a new space where the local constraints are better satisfied. While early methods were costly and could not be applied to medium-sized problems, recent methods offer better scalability and interesting features such as sparsity. Supervised metric learning from structured data has received less attention because it requires more complex procedures. Most of the work has focused on learning metrics based on the edit distance. Roughly speaking, the edit distance between two objects corresponds to the cheapest sequence of edit operations (insertion, deletion and substitution of subparts) turning one object into the other, where operations are assigned specific costs gathered in a matrix. Edit distance learning consists in optimizing the cost matrix and usually relies on maximizing the likelihood of pairs of similar examples in a probabilistic model.

Overall, we identify two main limitations of the current supervised metric learning methods. First, metrics are optimized based on *local* constraints and used in *local* algorithms, in particular k-nearest neighbors. However, it is unclear whether the same procedures can be used to obtain good metrics for use in *global* algorithms such as linear separators, which are simple yet powerful classifiers that often require less memory and provide greater prediction speed than k-nearest neighbors. In this context, one may want to optimize the metrics according to a *global* criterion but, to the best of our knowledge, this has never been addressed. Second, and perhaps more importantly, there is a substantial lack of theoretical understanding of generalization in metric learning. It is worth noting that in this context, the question of generalization is two-fold, as illustrated in Figure 1.1. First, one may be interested in the generalization ability of the metric itself, i.e., its consistency not only on the training sample but also on unseen data coming from the same distribution. Very little work has been done on this matter, and existing frameworks lack generality. Second, one may also be interested in the generalization ability of the learning algorithm that uses the learned metric, i.e., can we derive generalization guarantees for the learned model in terms of the quality of the learned metric? In practice, the learned metric is plugged into a learning algorithm and one can only hope that it yields good results. Although some approaches optimize the metric based on the decision rule of classification algorithms such as k-nearest neighbors, this question has never been investigated in a formal way. As we will see later in this document, the recently-proposed theory of (ϵ, γ, τ)-good similarity function (Balcan et al., 2008a,b) has been the first attempt to bridge the gap between the properties of a similarity function and its performance in linear classification, but has not been used so far in the context of metric learning. This theory plays a central role in two of our contributions.

The limitations described above constitute the main motivation for this thesis, and our contributions address them in several ways. First, we introduce a string kernel

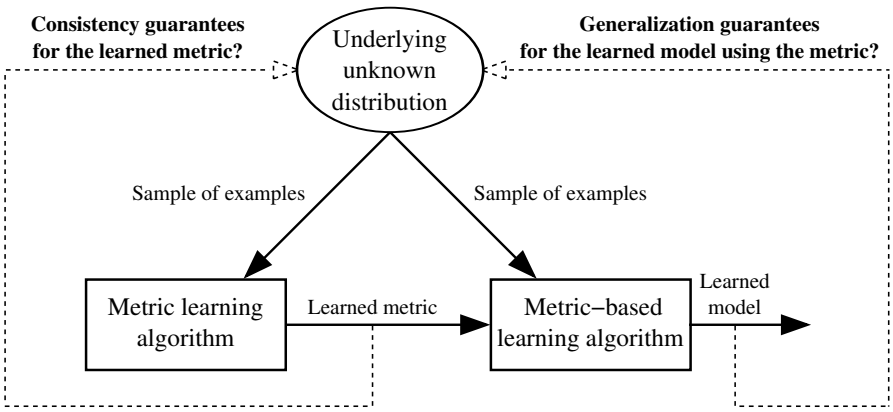


FIGURE 1.1: The two-fold problem of generalization in metric learning. We are interested in the generalization ability of the learned metric itself: can we say anything about its consistency on unseen data drawn from the same distribution? Furthermore, we are interested in the generalization ability of the learned model using that metric: can we relate its performance on unseen data to the quality of the learned metric?

that allows the use of learned edit distances in kernel-based methods such as support vector machines. This provides a way to use these learned metrics in global classifiers. Second, we propose two metric learning approaches based on (ϵ, γ, τ)-goodness, for which generalization guarantees can be derived both for the learned metric itself and for a linear classifier built from that metric. In the first approach (which deals with structured data), the metric is optimized with respect to local pairs to ensure the optimality of the solution. In the second approach, dealing with feature vectors allows us to optimize a global criterion that is more appropriate to linear classification. Lastly, we introduce a general framework that can be used to derive generalization guarantees for many existing metric learning methods based on local constraints.

Context of this work This thesis was carried out in the machine learning team of Laboratoire Hubert Curien UMR CNRS 5516, part of University of Saint-Étienne and University of Lyon. The contributions presented in this thesis were developed in the context of the ANR project Lampada[1] (ANR-09-EMER-007), which deals with scaling learning algorithms to handle large sets of structured data, with focuses on metric learning and sparse learning, and PASCAL2[2], a European Network of Excellence supporting research in machine learning, statistics and optimization.

Outline of the thesis This dissertation is organized as follows. Part I reviews the background work relevant to this thesis:

- Chapter 2 formally introduces the scientific context: supervised learning, analytical frameworks for deriving generalization guarantees, and various types of metrics.

[1] http://lampada.gforge.inria.fr/
[2] http://pascallin2.ecs.soton.ac.uk/

- Chapter 3 is a large survey of supervised metric learning from feature vectors and structured data, with a focus on the relative merits and limitations of the methods of the literature.

Part II gathers our contributions on metric learning from structured data:

- Chapter 4 introduces a new string kernel based on learned edit probabilities. Unlike other string edit kernels, it is parameter-free and guaranteed to be valid. Its naive form requires the computation of an infinite sum over all finite strings that can be built from the alphabet. We show how to get round this problem by using intersection of probabilistic automata and algebraic manipulation. Experiments highlight the performance of our kernel against state-of-the-art string kernels of the literature.

- Chapter 5 builds upon the theory of (ϵ, γ, τ)-good similarity function. We first show that we can use edit similarities directly in this framework and achieve competitive performance. The main contribution of this chapter is a novel method for learning string and tree edit similarities called GESL (for Good Edit Similarity Learning) that relies on a relaxed version of (ϵ, γ, τ)-goodness. The proposed approach, which is more flexible than previous methods, learn an edit similarity from *local* pairs and is then used to build a *global* linear classifier. Using uniform stability arguments, we are able to derive generalization guarantees for the learned similarity that actually give an upper bound on the generalization error of the linear classifier. We conduct extensive experiments that show the usefulness of our approach and the performance and sparsity of the resulting linear classifiers.

Part III gathers our contributions on metric learning from feature vectors:

- Chapter 6 presents a new bilinear similarity learning method for linear classification, called SLLC (for Similarity Learning for Linear Classification). Unlike GESL, SLLC directly optimizes the empirical (ϵ, γ, τ)-goodness criterion, which makes the approach entirely *global*: the similarity is optimized with respect to a global criterion (instead of local pairs) and plugged in a global linear classifier. SLLC is formulated as a convex minimization problem that can be efficiently solved in a batch or online way. We also kernelize our approach, thus learning a linear similarity in a nonlinear feature space induced by a kernel. Using similar arguments as for GESL, we derive generalization guarantees for SLLC highlighting that our method actually minimizes a tighter bound on the generalization error of the classifier than GESL. Experiments on several standard datasets show that SLLC leads to competitive classifiers that have the additional advantage of being very sparse, thus speeding up prediction.

- Chapter 7 addresses the lack of general framework for establishing generalization guarantees for metric learning. It is based on a simple adaptation of algorithmic robustness to the case where training data is made of pairs of examples. We show that a robust metric learning algorithm has generalization guarantees, and furthermore that a weak notion of robustness is actually necessary and sufficient for a metric learning algorithm to generalize. We illustrate the usefulness of our approach by showing that a large class of metric learning algorithms are robust. In particular, we are able to deal with sparsity-inducing regularizers, which was not possible with previous frameworks.

Notation Throughout this document, $\mathbb{N}$ denotes the set of natural numbers while $\mathbb{R}$ and $\mathbb{R}_+$ respectively denote the sets of real numbers and nonnegative real numbers. Arbitrary sets are denoted by calligraphic letters such as $\mathcal{S}$, and $|\mathcal{S}|$ stands for the number of elements in $\mathcal{S}$. A set of m elements from $\mathcal{S}$ is denoted by $\mathcal{S}^m$.

We denote vectors by bold lower case letters. For a vector $\mathbf{x} \in \mathbb{R}^d$ and $i \in [d] = \{1, \ldots, d\}$, x_i denotes the ith component of $\mathbf{x}$. The inner product between two vectors is denoted by $\langle \cdot, \cdot \rangle$. We denote matrices by bold upper case letters. For a $c \times d$ real-valued matrix $\mathbf{M} \in \mathbb{R}^{c \times d}$ and a pair of integers $(i, j) \in [c] \times [d]$, $M_{i,j}$ denotes the entry at row i and column j of the matrix $\mathbf{M}$. The identity matrix is denoted by $\mathbf{I}$ and the cone of symmetric positive semi-definite (PSD) $d \times d$ real-valued matrices by $\mathbb{S}^d_+$. $\| \cdot \|$ denotes an arbitrary (vector or matrix) norm and $\| \cdot \|_p$ the L_p norm. Strings are denoted by sans serif letters such as x. We use $|\mathsf{x}|$ to denote the length of x and $\mathsf{x_i}$ to refer to its ith symbol.

In the context of learning problems, we use $\mathcal{X}$ and $\mathcal{Y}$ to denote the input space (or instance space) and the output space (or label space) respectively. We use $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ to denote the joint space, and an arbitrary labeled instance is denoted by $z = (x, y) \in \mathcal{Z}$. The hinge function $[\cdot]_+ : \mathbb{R} \to \mathbb{R}_+$ is defined as $[c]_+ = \max(0, c)$. $\Pr[A]$ denotes the probability of the event A, $\mathbb{E}[X]$ the expectation of the random variable X and $x \sim P$ indicates that x is drawn according to the probability distribution P.

A summary of the notations is given in Table 1.1.

Notation	**Description**
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_+$	Set of nonnegative real numbers
$\mathbb{R}^d$	Set of d-dimensional real-valued vectors
$\mathbb{R}^{c \times d}$	Set of $c \times d$ real-valued matrices
$\mathbb{N}$	Set of natural numbers, i.e., $\{0, 1, \dots\}$
$\mathbb{S}_+^d$	Cone of symmetric PSD $d \times d$ real-valued matrices
$[k]$	The set $\{1, 2, \dots, k\}$
$\mathcal{S}$	An arbitrary set
$\lvert\mathcal{S}\rvert$	Number of elements in $\mathcal{S}$
$\mathcal{S}^m$	A set of m elements from $\mathcal{S}$
$\mathcal{X}$	Input space
$\mathcal{Y}$	Output space
$z = (x, y) \in \mathcal{X} \times \mathcal{Y}$	An arbitrary labeled instance
$\mathbf{x}$	An arbitrary vector
x_j, $x_{i,j}$	The jth component of $\mathbf{x}$ and $\mathbf{x_i}$
$\langle \cdot, \cdot \rangle$	Inner product between vectors
$[\cdot]_+$	Hinge function
$\mathbf{M}$	An arbitrary matrix
$\mathbf{I}$	The identity matrix
$M_{i,j}$	Entry at row i and column j of matrix $\mathbf{M}$
$\lVert \cdot \rVert$	An arbitrary norm
$\lVert \cdot \rVert_p$	L_p norm
x	An arbitrary string
$\lvert\mathsf{x}\rvert$	Length of string x
$\mathsf{x_i}, \mathsf{x_{i,j}}$	j^{th} symbol of x and $\mathsf{x_i}$
$x \sim P$	x is drawn i.i.d. from probability distribution P
$\Pr[\cdot]$	Probability of event
$\mathbb{E}[\cdot]$	Expectation of random variable

Table 1.1: Summary of notation.

PART I

Background

CHAPTER 2
Preliminaries

Chapter abstract In this chapter, we introduce the scientific context of this thesis as well as relevant background work. We first introduce formally the supervised learning setting and describe the main ideas of statistical learning theory, with a focus on binary classification. We then present three analytical frameworks (uniform convergence, uniform stability and algorithmic robustness) for establishing that a learning algorithm has generalization guarantees. Lastly, we recall the definition of several types of metrics and give examples of such functions for feature vectors and structured data.

2.1 Supervised Learning

The goal of supervised learning[1] is to automatically infer a model (hypothesis) from a set of labeled examples that is able to make predictions given new unlabeled data. In the following, we review basic notions of statistical learning theory, a very popular framework pioneered by Vapnik & Chervonenkis (1971). The interested reader can refer to Vapnik (1998) and Bousquet et al. (2003) for a more thorough description.

2.1.1 Typical Setting

In supervised learning, we learn a hypothesis from a set of labeled examples. This notion of training sample is formalized below.

Definition 2.1 (Training sample)**.** A training sample of size n is a set $\mathcal{T} = \{z_i = (x_i, y_i)\}_{i=1}^n$ of n observations independently and identically distributed (i.i.d.) according to an unknown joint distribution P over the space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ the output space. For a given observation z_i, $x_i \in \mathcal{X}$ is the instance (or example) and $y_i \in \mathcal{Y}$ its label. When $\mathcal{Y}$ is discrete, we are dealing with a classification task, and y_i is called the class of x_i. When $\mathcal{Y}$ is continuous, this is a regression task. In this thesis, we mainly focus on binary classification tasks, where we assume $\mathcal{Y} = \{-1, 1\}$.

[1] Note that there exist other learning paradigms, such as unsupervised learning (Ghahramani, 2003), semi-supervised learning (Chapelle et al., 2006), transfer learning (Pan & Yang, 2010), reinforcement learning (Sutton & Barto, 1998), etc.

We will mostly deal with feature vectors and strings. For feature vectors, we generally assume that $\mathcal{X} \subseteq \mathbb{R}^d$. For strings, we need the following definition.

Definition 2.2 (Alphabet and string)**.** An alphabet Σ is a finite nonempty set of symbols. A string x is a finite sequence of symbols from Σ. The empty string/symbol is denoted by \$ and Σ^* is the set of all finite strings (including \$) that can be generated from Σ. Finally, the length of a string x is denoted by $|\mathsf{x}|$.

We can now formally define what we mean by supervised learning.

Definition 2.3 (Supervised learning)**.** Supervised learning is the task of inferring a function (often referred to as a hypothesis or a model) $h_{\mathcal{T}} : \mathcal{X} \to \mathcal{L}$ belonging to some hypothesis class $\mathcal{H}$ from a training sample $\mathcal{T}$, which "best" predicts y from x for any (x, y) drawn from P. Note that the decision space $\mathcal{L}$ may or may not be equal to $\mathcal{Y}$.

In order to choose $h_{\mathcal{T}}$, we need a criterion to assess the quality of an arbitrary hypothesis h. Given a nonnegative loss function $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$ measuring the degree of agreement between $h(x)$ and y, we define the notion of true risk.

Definition 2.4 (True risk)**.** The true risk (also called generalization error) $R^\ell(h)$ of a hypothesis h with respect to a loss function ℓ is the expected loss suffered by h over the distribution P:

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

The most natural loss function for binary classification is the 0/1 loss (also called classification error):

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

$R^{\ell_{0/1}}(h)$ then corresponds to the proportion of time $h(x)$ and y agree in sign, and in particular to the proportion of correct predictions when $\mathcal{L} = \mathcal{Y}$.

The goal of supervised learning is then to find a hypothesis that achieves the smallest true risk. Unfortunately, in general we cannot compute the true risk of a hypothesis since the distribution P is unknown. We can only measure it empirically on the training sample. This is called the empirical risk.

Definition 2.5 (Empirical risk)**.** Let $\mathcal{T} = \{z_i = (x_i, y_i)\}_{i=1}^n$ be a training sample. The empirical risk (also called empirical error) $R^\ell_{\mathcal{T}}(h)$ of a hypothesis h over $\mathcal{T}$ with respect to a loss function ℓ is the average loss suffered by h on the instances in $\mathcal{T}$:

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

Under some restrictions, using the empirical risk to select the best hypothesis is a good strategy, as discussed in the next section.

2.1.2 Finding a Good Hypothesis

This section focuses on classic strategies for finding a good hypothesis in the true risk sense. The derivation of guarantees on the true risk of the selected hypothesis will be studied in Section 2.2.

Simply minimizing the empirical risk over all possible hypotheses would obviously be a good strategy if infinitely many training instances were available. Unfortunately, in realistic scenarios, training data is limited and there always exists a hypothesis h, however complex, that perfectly predicts the training sample, i.e., $R^{\ell}_{\mathcal{T}}(h) = 0$, but generalizes poorly, i.e., h has a nonzero (potentially large) true risk. This situation where the true risk of a hypothesis is much larger than its empirical risk is called *overfitting*. The intuitive idea behind it is that learning the training sample "by heart" does not provide good generalization to unseen data.

There is therefore a trade-off between minimizing the empirical risk and the complexity of the considered hypotheses, known as the bias-variance trade-off. There essentially exist two ways to deal with it and avoid overfitting: (i) restrict the hypothesis space, and (ii) favor simple hypotheses over complex ones. In the following, we briefly present three classic strategies for finding a hypothesis with small true risk.

Empirical Risk Minimization The idea of the Empirical Risk Minimization (ERM) principle is to pick a restricted hypothesis space $\mathcal{H} \subset \mathcal{L}^{\mathcal{X}}$ (for instance, linear classifiers, decision trees, etc.) and select a hypothesis $h_{\mathcal{T}} \in \mathcal{H}$ that minimizes the empirical risk:

$$h_{\mathcal{T}} = \underset{h \in \mathcal{H}}{\arg\min} \quad R^{\ell}_{\mathcal{T}}(h).$$

This may work well in practice but depends on the choice of hypothesis space. Essentially, we want $\mathcal{H}$ large enough to include hypotheses with small risk, but $\mathcal{H}$ small enough to avoid overfitting. Without background knowledge on the task, picking an appropriate $\mathcal{H}$ is difficult.

Structural Risk Minimization In Structural Risk Minimization (SRM), we use an infinite sequence of hypothesis classes $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \dots$ of increasing size and select the hypothesis that minimizes a penalized version of the empirical risk that favors "simple" classes:

$$h_{\mathcal{T}} = \underset{h \in \mathcal{H}_c, c \in \mathbb{N}}{\arg\min} \quad R^{\ell}_{\mathcal{T}}(h) + pen(\mathcal{H}_c).$$

Name	Formula	Pros	Cons
L_0 norm $\|\mathbf{x}\|_0$	Number of nonzero components	SP	NCO, NSM
L_1 norm $\|\mathbf{x}\|_1$	$\sum \|x_i\|$	CO, SP	NSM
(Squared) L_2 norm $\|\mathbf{x}\|_2^2$	$\sum x_i^2$	CO, SM	
$L_{2,1}$ norm $\|\mathbf{x}\|_{2,1}$	Sum of L_2 norms of grouped variables	CO, GSP	NSM

Table 2.1: Common regularizers on vectors. CO/NCO stand for convex/nonconvex, SM/NSM for smooth/nonsmooth and SP/GSP for sparsity/group sparsity.

Name	Formula	Pros	Cons
L_0 norm $\|\mathbf{M}\|_0$	Number of nonzero components	SP	NCO, NSM
L_1 norm $\|\mathbf{M}\|_1$	$\sum \|M_{i,j}\|$	CO, SP	NSM
(Squared) Frobenius norm $\|\mathbf{M}\|_{\mathcal{F}}^2$	$\sum M_{i,j}^2$	CO, SM	
$L_{2,1}$ norm $\|\mathbf{M}\|_{2,1}$	Sum of L_2 norms of rows/columns	CO, GSP	NSM
Trace (nuclear) norm $\|\mathbf{M}\|_*$	Sum of singular values	CO, LO	NSM

Table 2.2: Common regularizers on matrices. Abbreviations are the same as in Table 2.1, with LO standing for low-rank.

This implements the Occam's razor principle according to which one should choose the simplest explanation consistent with the training data.

Regularized Risk Minimization Regularized Risk Minimization (RRM) also builds upon the Occam's razor principle but is easier to implement: one picks a single, large hypothesis space $\mathcal{H}$ and a regularizer (usually some norm $\|h\|$) and selects a hypothesis that achieves the best trade-off between empirical risk minimization and regularization:

$$h_{\mathcal{T}} = \underset{h \in \mathcal{H}}{\arg\min} \quad R_{\mathcal{T}}^{\ell}(h) + \lambda \|h\|, \tag{2.1}$$

where λ is the trade-off parameter (in practice, it is set using validation data). The role of regularization is to penalize "complex" hypotheses. Note that it also provides a built-in way to break the tie between hypotheses that have the same empirical risk.

The choice of regularizer is important and depends on the considered task and the desired effect. Common regularizers for vector and matrix models are given in Table 2.1 and Table 2.2 respectively. Some regularizers are easy to optimize because they are convex and smooth (for instance, the squared L_2 norm) while others do not have these convenient properties and are thus harder to deal with (see Figure 2.1 for a graphical insight into some of these regularizers). However, the latter may bring some potentially interesting effects such as sparsity: they tend to set some parameters of the hypothesis to zero. Figure 2.2 illustrates this on L_2 and L_1 constraints — this also holds for regularization.[2]

[2]In fact, regularized and constrained problems are equivalent in the sense that for any value of the parameter β of a feasible constrained problem, there exists a value of the parameter λ of the

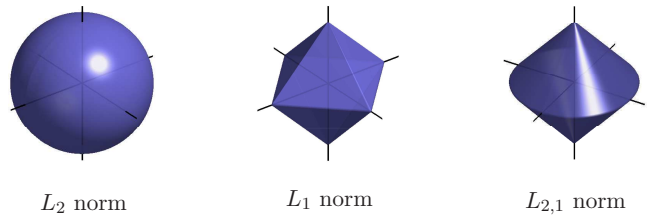


FIGURE 2.1: 3D unit balls of the L_1, L_2 and $L_{2,1}$ norms (taken from Grandvalet, 2011). The L_2 norm is convex, smooth and does not induce sparsity. The L_1 norm is convex, nonsmooth and induces sparsity at the coordinate level. The $L_{2,1}$ norm is convex, nonsmooth and induces sparsity at the group level (simultaneous sparsity of coordinates belonging to the same predefined group).

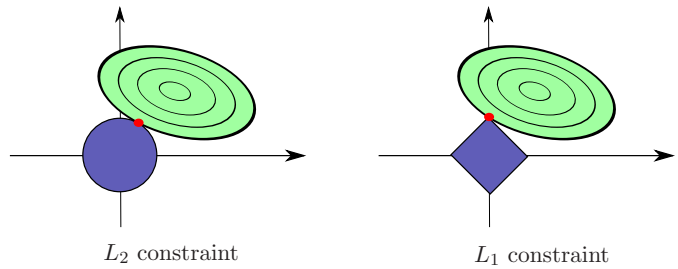


FIGURE 2.2: Geometric interpretation of L_2 and L_1 constraints in 2D. Suppose that we are looking for a hypothesis $\mathbf{h} \in \mathbb{R}^2$ with a constraint $\|\mathbf{h}\| \leq \beta$ (represented in dark blue) that minimizes the empirical risk (represented by the light green contour line). Unlike the L_2 norm, the L_1 norm tends to zero out coordinates, thus reducing dimensionality.

Regularization is used in many successful learning methods and, as we will see in Section 2.2, may help deriving generalization guarantees.

2.1.3 Surrogate Loss Functions

The methods described above all rely on minimizing the empirical risk. However, due to the nonconvexity of the 0/1 loss, minimizing (or approximately minimizing) $R^{\ell_{0/1}}$ is known to be NP-hard even for simple hypothesis classes (Ben-David et al., 2003). For this reason, surrogate convex loss functions (that can be more efficiently handled) are often used. The most prominent choices in the context of binary classification are:

- the hinge loss: $\ell_{hinge}(h, z) = [1 - yh(x)]_+ = \max(0, 1 - yh(x))$, used for instance in support vector machines (Cortes & Vapnik, 1995).
- the exponential loss: $\ell_{exp}(h, z) = e^{-yh(x)}$, used in Adaboost (Freund & Schapire, 1995).

corresponding regularized problem such that both problems have the same set of solutions, and vice versa. In practice, regularized problems are more convenient to use because they are always feasible.

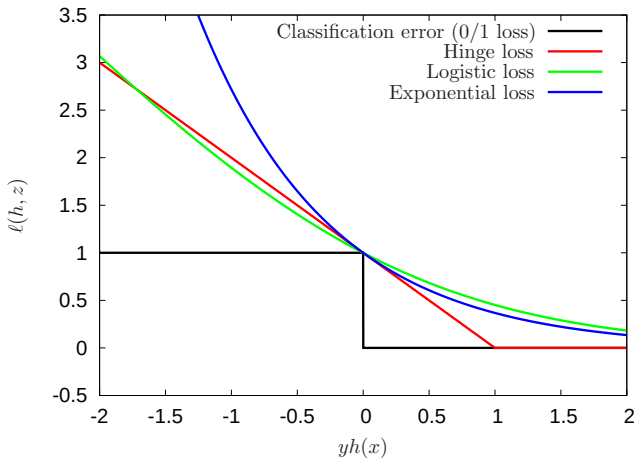


FIGURE 2.3: Plot of several loss functions for binary classification.

- the logistic loss: $\ell_{log}(h, z) = \log(1 + \epsilon^{-yh(x)})$, used in Logitboost (Friedman et al., 2000).

These loss functions are plotted in Figure 2.3 along with the nonconvex 0/1 loss.

Choosing an appropriate loss function is not an easy task and strongly depends on the problem, but there exist general results on the relative merits of different loss functions. For instance, Rosasco et al. (2004) studied statistical properties of several convex loss functions in a general classification setting and concluded that the hinge loss has a better convergence rate than other loss functions. Ben-David et al. (2012) have further shown that in the context of linear classification, the hinge loss offers the best guarantees in terms of classification error.

In the following section, we present analytical frameworks that allow the derivation of generalization guarantees, i.e., relating the empirical risk of $h_{\mathcal{T}}$ to its true risk.

2.2 Deriving Generalization Guarantees

In the previous section, we described a few generic methods for learning a hypothesis $h_{\mathcal{T}}$ from a training sample $\mathcal{T}$ based on minimizing the (penalized) empirical risk. However, learning a hypothesis with small *true* risk is what we are really interested in. Typically, the empirical risk can be seen as an optimistically biased estimation of the true risk (especially when the training sample is small), and a considerable amount of research has gone into deriving generalization guarantees for learning algorithms, i.e., bounding the deviation of the true risk of the learned hypothesis from its empirical measurement. These bounds are often referred to as PAC (Probably Approximately Correct) bounds

(Valiant, 1984) and have the following form:

$$\Pr[|R^{\ell}(h) - R^{\ell}_{\mathcal{T}}(h)| > \epsilon] \leq \delta,$$

where $\epsilon \geq 0$ and $\delta \in [0, 1]$. In other words, it bounds the probability to observe a large gap between the true risk and the empirical risk of an hypothesis.

The key instruments for deriving PAC bounds are concentration inequalities. They essentially assess the deviation of some functions of independent random variables from their expectation. Different concentration inequalities tackle different functions of the variables. The most commonly used in machine learning are Chebyshev (only one variable is considered), Hoeffding (sums of variables) and McDiarmid (that can accommodate any sufficiently regular function of the variables). For more details about concentration inequalities, see for instance the survey of Boucheron et al. (2004).

In this section, we present three theoretical frameworks for establishing generalization bounds: uniform convergence, uniform stability and algorithmic robustness (for a more general overview, please refer to the tutorial by Langford, 2005). Note that our contributions in Chapter 5, Chapter 6 and Chapter 7 make use of these frameworks.

2.2.1 Uniform Convergence

The theory of uniform convergence of empirical quantities to their mean (Vapnik & Chervonenkis, 1971; Vapnik, 1982) is one of the most prominent tools for deriving generalization bounds. It provides guarantees that hold for any hypothesis $h \in \mathcal{H}$ (including $h_{\mathcal{T}}$) and essentially bounds (with some probability $1 - \delta$) the true risk of h by its empirical risk plus a penalty term that depends on the number of training examples n, the size (or complexity) of the hypothesis space $\mathcal{H}$ and the value of δ. Intuitively, large n brings high confidence (since as $n \to \infty$ the empirical risk converges to the true risk by the law of large numbers), complex $\mathcal{H}$ brings low confidence (since overfitting is more likely), and δ accounts for the probability of drawing an "unlucky" training sample (i.e., not representative of the underlying distribution P).

When the hypothesis space is finite, we get the following PAC bound in $O(1/\sqrt{n})$.

Theorem 2.6 (Uniform convergence bound for the finite case)**.** *Let $\mathcal{T}$ be a training sample of size n drawn i.i.d. from some distribution P, $\mathcal{H}$ a finite hypothesis space and $\delta > 0$. For any $h \in \mathcal{H}$, with probability $1 - \delta$ over the random sample $\mathcal{T}$, we have:*

$$R^{\ell}(h) \leq R^{\ell}_{\mathcal{T}}(h) + \sqrt{\frac{\ln|\mathcal{H}| + \ln(1/\delta)}{2n}}.$$

When $\mathcal{H}$ is continuous (for instance, if $\mathcal{H}$ is the space of linear classifiers), we need a measure of the complexity of $\mathcal{H}$ such as the VC dimension (Vapnik & Chervonenkis,

1971), the fat-shattering dimension (Alon et al., 1997) or the Rademacher complexity (Koltchinskii, 2001; Bartlett & Mendelson, 2002). For instance, using the VC dimension, we get the following bound.

Theorem 2.7 (Uniform convergence bound with VC dimension)**.** *Let $\mathcal{T}$ be a training sample of size n drawn i.i.d. from some distribution P, $\mathcal{H}$ a continuous hypothesis space with VC dimension $VC(\mathcal{H})$ and $\delta > 0$. For any $h \in \mathcal{H}$, with probability $1-\delta$ over the random sample $\mathcal{T}$, we have:*

$$R^{\ell}(h) \leq R^{\ell}_{\mathcal{T}}(h) + \sqrt{\frac{VC(\mathcal{H})\left(\ln \frac{2n}{VC(\mathcal{H})} + 1\right) + \ln(4/\delta)}{n}}.$$

A drawback of uniform convergence analysis is that it is only based on the size of the training sample and the complexity of the hypothesis space, and completely ignores the learning algorithm, i.e., how the hypothesis $h_{\mathcal{T}}$ is selected.[3] In the following, we present two analytical frameworks that explicitly take into account the algorithm and can be used to derive generalization guarantees for $h_{\mathcal{T}}$ specifically, in particular in the regularized risk minimization setting (2.1).

2.2.2 Uniform Stability

Building on previous work on algorithmic stability, Bousquet & Elisseeff (2001, 2002) introduced new definitions that allow the derivation of generalization bounds for a large class of algorithms. Intuitively, an algorithm is said stable if it is robust to small changes in its input (in our case, the training sample), i.e., the variation in its output is small. Formally, we focus on uniform stability, a version of stability that allows the derivation of rather tight bounds.

Definition 2.8 (Uniform stability)**.** An algorithm $\mathcal{A}$ has uniform stability κ/n with respect to a loss function ℓ if the following holds:

$$\forall \mathcal{T}, |\mathcal{T}| = n, \forall i \in [n] : \sup_{z} |\ell(h_{\mathcal{T}}, z) - \ell(h_{\mathcal{T}^i}, z)| \leq \frac{\kappa}{n},$$

where κ is a positive constant, $\mathcal{T}^i$ is obtained from the training sample $\mathcal{T}$ by replacing the ith example $z_i \in \mathcal{T}$ by another example z'_i drawn i.i.d. from P, $h_{\mathcal{T}}$ and $h_{\mathcal{T}^i}$ are the hypotheses learned by $\mathcal{A}$ from $\mathcal{T}$ and $\mathcal{T}^i$ respectively.[4]

[3] In fact, the Rademacher complexity can sometimes implicitly take into account the regularization term of the algorithm.

[4] Definition 2.8 corresponds to the case where the training sample is altered through the replacement of an instance by another. Bousquet & Elisseeff (2001, 2002) also give a definition of uniform stability based on the removal of an instance from the training sample, which implies Definition 2.8. We will use Definition 2.8 throughout this thesis: we find it more convenient to deal with since replacement preserves the size of the training sample.

Bousquet & Elisseeff (2001, 2002) have shown that a large class of regularized risk minimization algorithms satisfies this definition. The constant κ typically depends on the form of the loss function, the regularizer and the regularization parameter λ. Making a good use of McDiarmid's inequality, they show that when Definition 2.8 is fulfilled, the following bound in $O(1/\sqrt{n})$ holds.

Theorem 2.9 (Uniform stability bound)**.** *Let $\mathcal{T}$ be a training sample of size n drawn i.i.d. from some distribution P and $\delta > 0$. For any algorithm $\mathcal{A}$ with uniform stability κ/n with respect to a loss function ℓ upper-bounded by some constant B,*[5] *with probability $1-\delta$ over the random sample $\mathcal{T}$, we have:*

$$R^{\ell}(h_{\mathcal{T}}) \leq R^{\ell}_{\mathcal{T}}(h_{\mathcal{T}}) + \frac{\kappa}{n} + (2\kappa + B)\sqrt{\frac{\ln(1/\delta)}{2n}},$$

where h_T is the hypothesis learned by $\mathcal{A}$ from $\mathcal{T}$.

The main difference between uniform convergence and uniform stability is that the latter incorporates regularization (through κ and h_T) and does not require any hypothesis space complexity argument. In particular, uniform stability can be used to derive generalization guarantees for hypothesis classes that are difficult to analyze with classic complexity arguments, such as k-nearest neighbors or support vector machines that have infinite VC dimension. It can also be adapted to non-i.i.d. settings (Mohri & Rostamizadeh, 2007, 2010). We will use uniform stability in the contributions presented in Chapter 5 and Chapter 6.

On the other hand, Xu et al. (2012a) have shown that algorithms with sparsity-inducing regularization are not stable.[6] Algorithmic robustness, presented in the next section, is able to deal with such algorithms. We will make use of this framework in Chapter 7.

2.2.3 Algorithmic Robustness

Algorithmic robustness (Xu & Mannor, 2010, 2012) is the ability of an algorithm to perform "similarly" on a training example and on a test example that are "close". It relies on a partitioning of the space $\mathcal{Z}$ to characterize closeness: two examples are close to each other if they lie in the same partition of the space. The partition itself is based on the notion of covering number (Kolmogorov & Tikhomirov, 1961).

Definition 2.10 (Covering number)**.** For a metric space $(\mathcal{S}, \rho)$ and $\mathcal{V} \subset \mathcal{S}$, we say that $\hat{\mathcal{V}} \subset \mathcal{V}$ is a γ-cover of $\mathcal{V}$ if $\forall t \in \mathcal{V}$, $\exists \hat{t} \in \hat{\mathcal{V}}$ such that $\rho(t, \hat{t}) \leq \gamma$. The γ-covering number of $\mathcal{V}$ is

$$\mathcal{N}(\gamma, \mathcal{V}, \rho) = \min\left\{|\hat{\mathcal{V}}| : \hat{\mathcal{V}} \text{ is a } \gamma\text{-cover of } \mathcal{V}\right\}.$$

[5] Note that many loss functions are unbounded if their domain is assumed to be unbounded (see Figure 2.3), but in practice they have bounded domain due for example to the common assumption that the norm of any instance is bounded.

[6] Sparsity is seen here as the ability to identify redundant features.

In particular, when $\mathcal{X}$ is compact, $\mathcal{N}(\gamma, \mathcal{X}, \rho)$ is finite, leading to a finite cover. Then, $\mathcal{Z}$ can be partitioned into $|\mathcal{Y}|\mathcal{N}(\gamma, \mathcal{X}, \rho)$ subsets such that if two examples $z = (x, y)$ and $z' = (x', y')$ belong to the same subset, then $y = y'$ and $\rho(x, x') \leq \gamma$.

We can now formally define the notion of robustness.

Definition 2.11 (Algorithmic robustness)**.** Algorithm $\mathcal{A}$ is $(K, \epsilon(\cdot))$-robust, for $K \in \mathbb{N}$ and $\epsilon(\cdot) : \mathcal{Z}^n \to \mathbb{R}$, if $\mathcal{Z}$ can be partitioned into K disjoint sets, denoted by $\{C_i\}_{i=1}^K$, such that the following holds for all $\mathcal{T} \in \mathcal{Z}^n$:

$$\forall z \in \mathcal{T}, \forall z' \in \mathcal{Z}, \forall i \in [K] : \text{if } z, z' \in C_i, \text{then } |\ell(h_\mathcal{T}, z) - \ell(h_\mathcal{T}, z')| \leq \epsilon(\mathcal{T}),$$

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

Briefly speaking, an algorithm is robust if for any example z' falling in the same subset as a training example z, then the gap between the losses associated with z and z' is bounded (by a quantity that may depend on the training sample $\mathcal{T}$). The existence of the partition itself is guaranteed by the definition of covering number. Note that both uniform stability and algorithmic robustness properties involve a bound on deviations between losses. The key difference is that uniform stability studies the variation of the loss associated with any example z under small changes in the training sample (implying that the learned hypothesis itself does not vary much), while algorithmic robustness considers the deviation between the losses associated with two examples that are close (implying that the learned hypothesis is locally consistent).

Xu & Mannor (2010, 2012) have shown that a robust algorithm has generalization guarantees. This is formalized by the following theorem.

Theorem 2.12 (Robustness bound)**.** *Let ℓ be a loss function upper-bounded by some constant B, and $\delta > 0$. If an algorithm $\mathcal{A}$ is $(K, \epsilon(\cdot))$-robust, then with probability $1 - \delta$, we have:*

$$R^\ell(h_\mathcal{T}) \leq R^\ell_\mathcal{T}(h_\mathcal{T}) + \epsilon(\mathcal{T}) + B\sqrt{\frac{2K \ln 2 + 2\ln(1/\delta)}{n}},$$

where h_T is the hypothesis learned by $\mathcal{A}$ from $\mathcal{T}$.

Note that there is a tradeoff between the size K of the partition and $\epsilon(\mathcal{T})$: the latter can essentially be made as small as possible by using a finer-grained cover.

PAC bounds based on robustness are generally not tight since they rely on unspecified (potentially large) covering numbers. On the other hand, a great advantage of robustness is that it can deal with a larger class of regularizers than stability (in particular, sparsity-inducing norms can be considered), and its geometric interpretation makes adaptations to non-standard settings (such as non-i.i.d. data) possible. Our contribution in Chapter 7 adapts robustness to the case of metric learning, when training data consist of non-i.i.d. pairs of examples. Finally, note that Xu & Mannor (2010, 2012)

established that a weak notion of robustness is necessary and sufficient for an algorithm to generalize asymptotically, making robustness a key property for the generalization of learning algorithms.

After having presented the supervised learning setting and analytical frameworks for deriving generalization guarantees, we now turn to the topic of metrics, which has a great place in this thesis.

2.3 Metrics

The notion of metric (used here as a generic term for distance, similarity or dissimilarity function) plays an important role in many machine learning problems such as classification, regression, clustering, or ranking. Successful examples include:

- k-Nearest Neighbors (k-NN) classification (Cover & Hart, 1967), where the predicted class of an instance x corresponds to the majority class among the k-nearest neighbors of x in the training sample, according to some distance or similarity.
- Kernel methods (Schölkopf & Smola, 2001), where a specific type of similarity function called kernel (see Definition 2.15) is used to implicitly project data into a new high-dimensional feature space. The most prominent example is Support Vector Machines (SVM) classification (Cortes & Vapnik, 1995), where a large-margin linear classifier is learned in that space.
- K-Means (Lloyd, 1982), a clustering algorithm which aims at finding the K clusters that minimize the within-cluster distance on the training sample according to some metric.
- Information retrieval, where a similarity function is often used to retrieve documents (webpages, images, etc.) that are similar to a query or to another document (Salton et al., 1975; Baeza-Yates & Ribeiro-Neto, 1999; Sivic & Zisserman, 2009).
- Data visualization, where visualization of interesting patterns in high-dimensional data is sometimes achieved by means of a metric (Venna et al., 2010; Bertini et al., 2011).

It should be noted that metrics are especially important when dealing with structured data (such as strings, trees, or graphs) because they are often a convenient proxy to manipulate these complex objects: if a metric is available, then any metric-based algorithm (such as those presented in the above list) can be used.

In this section, we first give the definitions of distance, similarity and kernel functions (2.3.1), and then give some examples (by no means an exhaustive list) of such metrics between feature vectors (2.3.2) and between structured data (2.3.3).

2.3.1 Definitions

We start by introducing the definition of a distance function.

Definition 2.13 (Distance function)**.** A distance over a set $\mathcal{X}$ is a pairwise function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which satisfies the following properties $\forall x, x', x'' \in \mathcal{X}$:

1. $d(x, x') \geq 0$ (nonnegativity),
2. $d(x, x') = 0$ if and only if $x = x'$ (identity of indiscernibles),
3. $d(x, x') = d(x', x)$ (symmetry),
4. $d(x, x'') \leq d(x, x') + d(x', x'')$ (triangle inequality).

A *pseudo-distance* satisfies the properties of a metric, except that instead of property 2, only $d(x, x) = 0$ is required. Note that the property of triangle inequality can be used to speedup learning algorithms such as k-NN (e.g., Micó et al., 1994; Lai et al., 2007; Wang, 2011) or K-Means (Elkan, 2003).

While a distance function is a well-defined mathematical concept, there is no general agreement on the definition of a (dis)similarity function, which can essentially be any pairwise function. Throughout this thesis, we will use the following definition.

Definition 2.14 (Similarity function)**.** A (dis)similarity function is a pairwise function $K : \mathcal{X} \times \mathcal{X} \to [-1, 1]$. We say that K is a symmetric similarity function if $\forall x, x' \in \mathcal{X}$, $K(x, x') = K(x', x)$.

A similarity function should return a high score for similar inputs and a low score for dissimilar ones (the other way around for a dissimilarity function). Note that (normalized) distance functions are dissimilarity functions.

Finally, a kernel is a special type of similarity function, as formalized by the following definition.

Definition 2.15 (Kernel function)**.** A symmetric similarity function K is a kernel if there exists a (possibly implicit) mapping function $\phi : \mathcal{X} \to \mathbb{H}$ from the instance space $\mathcal{X}$ to a Hilbert space $\mathbb{H}$ such that K can be written as an inner product in $\mathbb{H}$:

$$K(x, x') = \langle \phi(x), \phi(x') \rangle .$$

Equivalently, K is a kernel if it is positive semi-definite (PSD), i.e.,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$$

for all finite sequences of $x_1, \ldots, x_n \in \mathcal{X}$ and $c_1, \ldots, c_n \in \mathbb{R}$.

Kernel functions are a key component of kernel methods such as SVM, because they can implicitly allow cheap inner product computations in very high-dimensional spaces (this is known as the "kernel trick") and bring an elegant theory based on Reproducing Kernel Hilbert Spaces (RKHS). Note that these advantages disappear when using an arbitrary non-PSD similarity function instead of a kernel, and the convergence of the kernel-based algorithm may not even be guaranteed in this case.[7]

2.3.2 Some Metrics between Feature Vectors

Minkowski distances Minkowski distances are a family of distances induced by L_p norms. For $p \geq 1$,

$$d_p(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_p = \left(\sum_{i=1}^{d} |x_i - x_i'|^p \right)^{1/p}. \tag{2.2}$$

From (2.2) we can recover three widely used distances:

- When $p = 1$, we get the Manhattan distance:

$$d_{man}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_1 = \sum_{i=1}^{d} |x_i - x_i'|.$$

- When $p = 2$, we get the "ordinary" Euclidean distance:

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

- When $p \to \infty$, we get the Chebyshev distance:

$$d_{che}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_\infty = \max_i |x_i - x_i'|.$$

Note that when $0 < p < 1$, d_p is not a proper distance (it violates the triangle inequality) and the corresponding (pseudo) norm is nonconvex. Figure 2.4 shows the corresponding unit circles for several values of p.

Mahalanobis distances The Mahalanobis distance, which incorporates knowledge about the correlation between features, is defined by

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

[7] Some research has gone into training SVM with indefinite kernels, mostly based on building a PSD kernel from the indefinite one while learning the SVM classifier. The interested reader may refer to the work of Ong et al. (2004); Luss & d'Aspremont (2007); Chen & Ye (2008); Chen et al. (2009) and references therein.

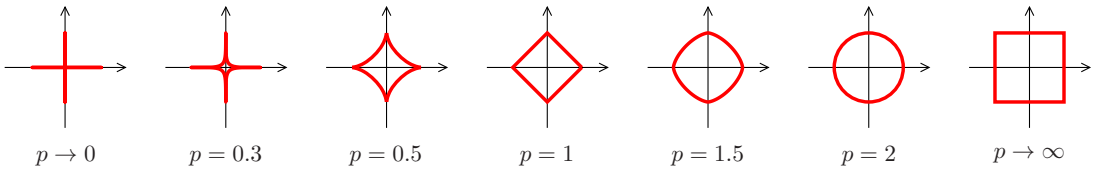


FIGURE 2.4: Minkowski distances: unit circles for various values of p.

where $\mathbf{x}$ and $\mathbf{x}'$ are random vectors from the same distribution with covariance matrix $\mathbf{\Sigma}$. The term Mahalanobis distance is also used to refer to the following generalization of the original definition, sometimes referred to as generalized quadratic distances (Nielsen & Nock, 2009):

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

where $\mathbf{M} \in \mathbb{S}_+^d$. $\mathbb{S}_+^d$ denotes the cone of symmetric PSD $d \times d$ real-valued matrices. $\mathbf{M} \in \mathbb{S}_+^d$ ensures that $d_{\mathbf{M}}$ is a pseudo-distance. When $\mathbf{M}$ is the identity matrix, we recover the Euclidean distance. Otherwise, using Cholesky decomposition, one can rewrite $\mathbf{M}$ as $\mathbf{L}^T\mathbf{L}$, where $\mathbf{L} \in \mathbb{R}^{k \times d}$, where k is the rank of $\mathbf{M}$. Hence:

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

Thus, a Mahalanobis distance implicitly corresponds to computing the Euclidean distance after the linear projection of the data defined by $\mathbf{L}$. Note that if $\mathbf{M}$ is low-rank, i.e., $\text{rank}(\mathbf{M}) = r < d$, then it induces a linear projection of the data into a space of lower dimension r. It thus allows a more compact representation of the data and cheaper distance computations, especially when the original feature space is high-dimensional.

Because of these nice properties, learning Mahalanobis distance has attracted a lot of interest and is a major component of metric learning (see Section 3.2.1).

Cosine similarity The cosine similarity measures the cosine of the angle between two instances, and can be computed as

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

The cosine similarity is widely used in data mining, in particular in text retrieval (Baeza-Yates & Ribeiro-Neto, 1999) and more recently in image retrieval (see for instance Sivic & Zisserman, 2009) when data are represented as term vectors (Salton et al., 1975).

Bilinear similarity The bilinear similarity is related to the cosine similarity but does not include normalization by the norms of the inputs and is parameterized by a matrix $\mathbf{M}$:

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

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is not required to be PSD nor symmetric. The bilinear similarity has been used for instance in image retrieval (Deng et al., 2011). When $\mathbf{M}$ is the identity matrix, $K_{\mathbf{M}}$ amounts to an unnormalized cosine similarity. The bilinear similarity has two advantages. First, it is efficiently computable for sparse inputs: if $\mathbf{x}$ and $\mathbf{x}'$ have k_1 and k_2 nonzero features, $K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$ can be computed in $O(k_1 k_2)$ time. Second, unlike Minkowski distance, Mahalanobis distances and the cosine similarity, it can be easily used as a similarity measure between instances of different dimension (for example, a document and a query) by choosing a nonsquare matrix $\mathbf{M}$. A major contribution of this thesis is to propose a novel method for learning a bilinear similarity (Chapter 6).

Linear kernel The linear kernel is simply the inner product in the original space $\mathcal{X}$:

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

In other words, the corresponding ϕ is an identity map: $\forall \mathbf{x} \in \mathcal{X}, \phi(\mathbf{x}) = \mathbf{x}$. Note that K_{lin} corresponds to the bilinear similarity with $\mathbf{M} = \mathbf{I}$.

Polynomial kernels Polynomial kernels are defined as:

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

where $deg \in \mathbb{N}$. It can be shown that K_{deg} implicitly projects an instance into the nonlinear space $\mathbb{H}$ of all monomials of degree up to deg.

Gaussian kernel The Gaussian kernel, also known as the RBF kernel, is a widely used kernel defined by

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

where $\sigma^2 > 0$ is a width parameter. For this kernel, it can be shown that the corresponding implicit nonlinear projection space $\mathbb{H}$ is infinite-dimensional.

2.3.3 Some Metrics between Structured Data

Hamming distance The Hamming distance is a distance between strings of identical length and is equal to the number of positions at which the symbols differ. It has been

C	$	a	b
$	0	2	10
a	2	0	4
b	10	4	0

TABLE 2.3: Example of edit cost matrix **C**. Here, $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

used mostly for binary strings and is defined by

$$d_{ham}(\mathsf{x}, \mathsf{x}') = |\{i : \mathsf{x}_\mathsf{i} \neq \mathsf{x}'_\mathsf{i}\}|.$$

String edit distance The string edit distance (Levenshtein, 1966) is a distance between strings of possibly different length built from an alphabet Σ. It is based on three elementary edit operations: insertion, deletion and substitution of a symbol. In the more general version, each operation has a specific cost, gathered in a nonnegative $(|\Sigma| + 1) \times (|\Sigma| + 1)$ matrix **C** (the additional row and column account for insertion and deletion costs respectively). A sequence of operations transforming a string x into a string x' is called an edit script. The edit distance between x and x' is defined as the cost of the cheapest edit script that turns x into x' and can be computed in $O(|\mathsf{x}| \cdot |\mathsf{x}'|)$ time by dynamic programming.[8]

The classic edit distance, known as the Levenshtein distance, uses a unit cost matrix and thus corresponds to the minimum number of operations turning one string into another. For instance, the Levenshtein distance between `abb` and `aa` is equal to 2, since turning `abb` into `aa` requires at least 2 operations (e.g., substitution of `b` with `a` and deletion of `b`). On the other hand, using the cost matrix given in Table 2.3, the edit distance between `abb` and `aa` is equal to 10 (deletion of `a` and two substitutions of `b` with `a` is the cheapest edit script).

Using task-specific costs is a key ingredient to the success of the edit distance in many applications. For some problems such as handwritten character recognition (Micó & Oncina, 1998) or protein alignment (Dayhoff et al., 1978; Henikoff & Henikoff, 1992), relevant cost matrices may be available. But a more general solution consists in automatically learning the cost matrix from data, as we shall see in Section 3.3.1. One of the contributions of this thesis is to propose a new edit cost learning method (Chapter 5).

Sequence alignment Sequence alignment is a way of computing the similarity between two strings, mostly used in bioinformatics to identify regions of similarity in DNA or protein sequences (Mount, 2004). It corresponds to the score of the best alignment. The score of an alignment is based on the same elementary operations as the

[8] Note that in the case of strings of equal length, the edit distance is upper bounded by the Hamming distance.

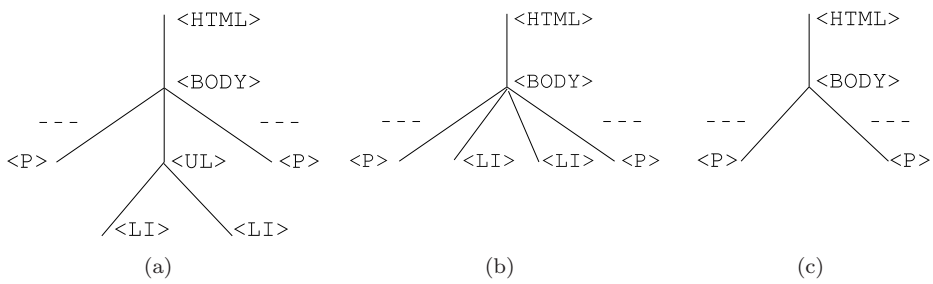


FIGURE 2.5: Strategies to delete a node within a tree: (a) original tree, (b) after deletion of the `<UL>` node as defined by Zhang & Shasha, and (c) after deletion of the `<UL>` node as defined by Selkow.

edit distance and on a score matrix for substitutions, but uses a (linear or affine) gap penalty function instead of insertion and deletion costs. The most prominent sequence alignment measures are the Needleman-Wunsch score (Needleman & Wunsch, 1970) for global alignments and the Smith-Waterman score (Smith & Waterman, 1981) for local alignments. They can be computed by dynamic programming.

Tree edit distance Because of the growing interest in applications that naturally involve tree-structured data (such as the secondary structure of RNA in biology, XML documents on the web or parse trees in natural language processing), several works have extended the string edit distance to trees, resorting to the same elementary edit operations (see Bille, 2005, for a survey on the matter). There exist two main variants of the tree edit distance that differ in the way the deletion of a node is handled. In Zhang & Shasha (1989), when a node is deleted all its children are connected to its father. The best algorithms for computing this distance have an $O(n^3)$ worst-case complexity, where n is the number of nodes of the largest tree (see Pawlik & Augsten, 2011, for an empirical evaluation of several algorithms). Another variant is due to Selkow (1977), where insertions and deletions are restricted to the leaves of the tree. Such a distance is relevant to specific applications. For instance, deleting a `<UL>` tag (i.e., a nonleaf node) of an unordered list in an HTML document would require the iterative deletion of the `<LI>` items (i.e., the subtree) first, which is a sensible thing to do in this context (see Figure 2.5). This version can be computed in quadratic time. Note that tree edit distance computations can be made significantly faster (especially for large trees) by exploiting lower bounds on the distance between two trees that are cheap to obtain (see for instance Yang et al., 2005). A study on the expressiveness of similarities and distances on trees was proposed by Emms & Franco-Penya (2012).

Like in the string case, there exists a few methods for learning the cost matrix of the tree edit distance (see Section 3.3.2). Note that our edit similarity learning method, presented in Chapter 5, can be used for both strings and trees.

Graph edit distance Note that there also exist extensions of the edit distance to general graphs (Gao et al., 2010), but like many problems on graphs, computing a graph edit distance is NP-hard, making it impractical for real-world tasks.

Spectrum, subsequence and mismatch kernels These string kernels represent strings by fixed-length feature vectors and rely on explicit mapping functions ϕ. The spectrum kernel (Leslie et al., 2002a) maps each string to a vector of frequencies of all contiguous subsequences of length p and computes the inner product between these vectors. The subsequence kernel (Lodhi et al., 2002) and the mismatch kernel (Leslie et al., 2002b) extend the spectrum kernel to inexact subsequence matching: the former considers all (possibly noncontiguous) subsequences of length p while the latter allows a number of mismatches in the subsequences.

String edit kernels String edit kernels are derived from the string edit distance (or related measures). The classic edit kernel (Li & Jiang, 2004) has the following form:

$$K_{L\&J}(\mathsf{x}, \mathsf{x}') = e^{-t \cdot d_{lev}(\mathsf{x}, \mathsf{x}')},$$

where d_{lev} is the Levenshtein distance and $t > 0$ is a parameter. However, Cortes et al. (2004) have shown that this function is not PSD (and thus is not a valid kernel) in the general case for nontrivial alphabets. Thus, one has to tune t, hoping to make K PSD. Moreover, it suffers from the so-called "diagonal dominance" problem (i.e., the kernel value decreases exponentially fast with the distance), and SVM is known not to perform well in this case (Schölkopf et al., 2002). A different string edit kernel was proposed by Neuhaus & Bunke (2006) and is defined as follows:

$$K_{N\&B}(\mathsf{x}, \mathsf{x}') = \frac{1}{2}(d_{lev}(\mathsf{x}, \mathsf{x}_0)^2 + d_{lev}(\mathsf{x}_0, \mathsf{x}')^2 - d_{lev}(\mathsf{x}, \mathsf{x}')^2),$$

where x_0 is called the "zero string" and must be picked by hand. They also propose combinations of such kernels with different zero strings. However, the validity of such kernels is not guaranteed either. Saigo et al. (2004) build a kernel from the sum of scores over all possible Smith-Waterman local alignments between two strings instead of the alignment of highest score only. They show that if the score matrix is PSD, then the kernel is valid in general. However, like $K_{L\&J}$, it suffers from the diagonal dominance problem. In practice, the authors take the logarithm of the kernel and add a sufficiently large diagonal term to ensure the validity of the kernel.

Convolution kernels The framework of convolution kernels (Haussler, 1999) can be used to derive many kernels for structured data. Roughly speaking, if structured instances can be seen as a collection of subparts, then Haussler's convolution kernel between two instances is defined as the sum of the return values of a predefined kernel

over all possible pairs of subparts, and is guaranteed to be PSD. Mapping kernels (Shin & Kuboyama, 2008) are a generalization of convolution kernels as they allow the sum to be computed only over a predefined subset of the subpart pairs. These frameworks have been used to design several kernels between structured data (Collins & Duffy, 2001; Shin & Kuboyama, 2008; Shin et al., 2011). However, building such kernels is often not straightforward since they suppose the existence of a kernel between subparts of the structured instances.

Marginalized kernels When one has access to a probabilistic model encoding for instance the probability that a string (or a tree) is turned into another one, marginalized kernels (Tsuda et al., 2002; Kashima et al., 2003), of which the Fisher kernel (Jaakkola & Haussler, 1998) is a special case, are a way of building a kernel from the output of such models. Since our string kernel proposed in Chapter 4 belongs to this family, we postpone the details of the framework to Section 4.2.

2.4 Conclusion

In this chapter, we introduced the setting of supervised learning, presented analytical frameworks that allow the derivation of generalization bounds for learning algorithms, and reviewed different forms of metrics.

The contributions of this thesis can be cast as supervised metric learning methods, i.e., learning the parameters of a metric from labeled data. Because the performance of many learning algorithms using metrics critically depends on the relevance of the metric to the problem at hand, supervised metric learning has attracted a lot of interest in recent years. Chapter 3 is a large review of the literature on the subject.

CHAPTER 3
A Review of Supervised Metric Learning

Chapter abstract In this chapter, we review the literature on supervised metric learning. We start by introducing the main concepts of this research topic. Then, we cover metric learning from feature vectors (in particular, Mahalanobis distance learning) as well as metric learning from structured data such as strings and trees, with an emphasis on the pros and cons of each method. Finally, we conclude by discussing the general limitations of the current literature that motivate our work.

3.1 Introduction

As discussed in Section 2.3, using an appropriate metric is key to the performance of many learning algorithms. Since manually tuning metrics (when they allow some parameterization) for a given real-world problem is often difficult and tedious, a lot of work has gone into automatically learning them from labeled data, leading to the emergence of metric learning. This chapter is devoted to a large survey of supervised metric learning techniques.

Generally speaking, supervised metric learning approaches rely on the reasonable intuition that a good similarity function should assign a large (resp. small) score to pairs of points of the same class (resp. different class), and conversely for a distance function. Following this idea, they aim at finding the parameters (usually a matrix) of the metric such that it best satisfies local constraints built from the training sample $\mathcal{T}$. They are typically pair or triplet-based constraints of the following form:

$$\begin{aligned}
\mathcal{S} &= \{(z_i, z_j) \in \mathcal{T} \times \mathcal{T} : x_i \text{ and } x_j \text{ should be similar}\}, \\
\mathcal{D} &= \{(z_i, z_j) \in \mathcal{T} \times \mathcal{T} : x_i \text{ and } x_j \text{ should be dissimilar}\}, \\
\mathcal{R} &= \{(z_i, z_j, z_k) \in \mathcal{T} \times \mathcal{T} \times \mathcal{T} : x_i \text{ should be more similar to } x_j \text{ than to } x_k\},
\end{aligned}$$

where $\mathcal{S}$ and $\mathcal{D}$ are often referred to as the positive and negative training pairs respectively, and $\mathcal{R}$ as the training triplets. These constraints are usually derived from the labels of the training instances. One may consider for instance all possible pairs/triplets

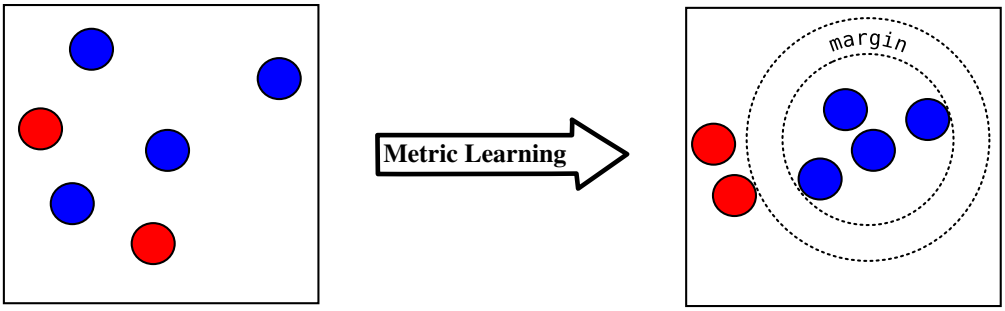


FIGURE 3.1: Intuition behind metric learning. Before learning (left pane), red and blue points are not well-separated. After learning (right pane), red and blues points are separated by a certain margin.

or use only a subset of these, for instance based on random selection or a notion of neighborhood.

Metric learning often has a geometric interpretation: it can be seen as finding a new feature space for the data where the local constraints are better satisfied (see Figure 3.1 for an example). Learned metrics are typically used to improve the performance of learning algorithms based on local neighborhoods such as k-NN.

The rest of this chapter is organized as follows. Section 3.2 reviews metric learning approaches where data consist of feature vectors while Section 3.3 deals with metric learning from structured data. We conclude with a summary of the main features of the studied approaches and a discussion on some of their limitations in Section 3.4.

3.2 Metric Learning from Feature Vectors

In this section, we focus on metric learning methods for data lying in some feature space $\mathcal{X} \subseteq \mathbb{R}^d$. In Section 3.2.1, we review Mahalanobis distance learning, which has attracted most of the interest, as well as similarity learning in Section 3.2.2 and nonlinear metric learning in Section 3.2.3. Finally, we list a few approaches designed for other settings in Section 3.2.4.

3.2.1 Mahalanobis Distance Learning

A great deal of work has focused on learning a (squared) Mahalanobis distance $d^2_{\mathbf{M}}$ parameterized by $\mathbf{M} \in \mathbb{S}^d_+$. Maintaining $\mathbf{M} \in \mathbb{S}^d_+$ in an efficient way during the optimization process is a key challenge in Mahalanobis distance learning. Indeed, general Semi-Definite Programming (SDP) techniques (Vandenberghe & Boyd, 1996), i.e., optimization over the PSD cone, consists in repeatedly performing a gradient step on the objective function followed by a projection step onto the PSD cone (which is done by

setting the negative eigenvalues to zero). This is slow in practice because it requires eigenvalue decomposition, which scales in $O(d^3)$. Another interesting challenge is to learn a low-rank matrix (which implies a low-dimensional projection space, as noted earlier) instead of a full-rank one, since optimizing $\mathbf{M}$ subject to a rank constraint or regularization is NP-hard and thus cannot be carried out efficiently.

In this section, we review the main supervised Mahalanobis distance learning methods of the literature. We first present two early approaches that deal with the PSD constraint in a rudimentary way (Section 3.2.1.1). We then discuss approaches that are specific to k-nearest neighbors (Section 3.2.1.2), inspired from information theory (Section 3.2.1.3), online learning methods (Section 3.2.1.4), approaches with generalization guarantees (Section 3.2.1.5) and a few more that do not fit any of the previous categories (Section 3.2.1.6).

3.2.1.1 Early Approaches

MMC (Xing et al.) The pioneering work of Xing et al. (2002) is the first Mahalanobis distance learning method. It relies on a convex SDP formulation with no regularization, which aims at maximizing the sum of distances between dissimilar points while keeping the sum of distances between similar examples small:

$$\begin{aligned} \max_{\mathbf{M}\in\mathbb{S}_+^d} \quad & \sum_{(z_i,z_j)\in\mathcal{D}} d_{\mathbf{M}}(\mathbf{x_i},\mathbf{x_j}) \\ \text{s.t.} \quad & \sum_{(z_i,z_j)\in\mathcal{S}} d^2_{\mathbf{M}}(\mathbf{x_i},\mathbf{x_j}) \leq 1. \end{aligned} \tag{3.1}$$

The algorithm for solving (3.1) is a basic SDP approach based on eigenvalue decomposition. This makes it intractable for medium and high-dimensional problems.

Schultz & Joachims The method proposed by Schultz & Joachims (2003) relies on the assumption that $\mathbf{M} = \mathbf{A}^T\mathbf{W}\mathbf{A}$, where $\mathbf{A}$ is fixed and known and $\mathbf{W}$ is diagonal. We get:

$$d^2_{\mathbf{M}}(\mathbf{x_i},\mathbf{x_j}) = (\mathbf{A}\mathbf{x_i} - \mathbf{A}\mathbf{x_j})^T\mathbf{W}(\mathbf{A}\mathbf{x_i} - \mathbf{A}\mathbf{x_j}).$$

By definition, $\mathbf{M}$ is PSD and thus one can optimize over the diagonal matrix $\mathbf{W}$ and avoid the need for SDP. They propose a formulation based on triplet constraints:

$$\begin{aligned} \min_{\mathbf{W}} \quad & \|\mathbf{M}\|^2_{\mathcal{F}} \\ \text{s.t.} \quad & d^2_{\mathbf{M}}(\mathbf{x_i},\mathbf{x_k}) - d^2_{\mathbf{M}}(\mathbf{x_i},\mathbf{x_j}) \geq 1 \quad \forall (z_i,z_j,z_k)\in\mathcal{R}, \end{aligned} \tag{3.2}$$

where $\|\cdot\|^2_{\mathcal{F}}$ is the squared Frobenius norm. Slack variables are introduced to allow soft constraints. Problem (3.2) is convex and can be solved efficiently. The main drawback

of this approach is that it is less general than full Mahalanobis distance learning: one only learns a weighting $\mathbf{W}$ of the features. Furthermore, $\mathbf{A}$ must be chosen manually.

3.2.1.2 Approaches driven by Nearest Neighbors

The objective functions of the methods presented in this section are related to a nearest neighbor prediction rule.

NCA (Goldberger et al.) The idea of Neighborhood Component Analysis (NCA), introduced by Goldberger et al. (2004), is to optimize the expected leave-one-out error of a stochastic nearest neighbor classifier in the projection space induced by $d_{\mathbf{M}}$. They use the decomposition $\mathbf{M} = \mathbf{L}^T\mathbf{L}$ and they define the probability that $\mathbf{x_i}$ is the neighbor of $\mathbf{x_j}$ by

$$p_{ij} = \frac{\exp(-\|\mathbf{Lx_i} - \mathbf{Lx_j}\|^2)}{\sum_{l \neq i} \exp(-\|\mathbf{Lx_i} - \mathbf{Lx_l}\|^2)}, \quad p_{ii} = 0.$$

Then, the probability that $\mathbf{x_i}$ is correctly classified is:

$$p_i = \sum_{j:y_j=y_i} p_{ij}.$$

They learn the distance by solving:

$$\max_L \sum_i p_i. \tag{3.3}$$

Note that the matrix $\mathbf{L}$ can be chosen nonsquare, inducing a low-rank $\mathbf{M}$. The main limitation of (3.3) is that it is nonconvex and thus subject to local maxima.

MCML (Globerson & Roweis) Later on, Globerson & Roweis (2005) proposed an alternative convex formulation based on minimizing a KL divergence between p_{ij} and an ideal distribution. Unlike NCA, this is done with respect to the matrix $\mathbf{M}$. However, like MMC, MCML requires costly projections onto the PSD cone.

LMNN (Weinberger et al.) Large Margin Nearest Neighbors (LMNN), introduced by Weinberger et al. (2005; 2008; 2009), is one of the most popular Mahalanobis distance learning methods. The idea is to learn the distance such that the k nearest neighbors belong to the correct class while keeping away instances of other classes. The Euclidean distance is used to determine these "target neighbors". Formally, the constraints are

defined in the following way:

$$\begin{aligned}\mathcal{S} &= \{(z_i, z_j) \in \mathcal{T} \times \mathcal{T} : \ell_i = \ell_j \text{ and } \mathbf{x_j} \text{ belongs to the } k\text{-neighborhood of } \mathbf{x_i}\},\\ \mathcal{R} &= \{(z_i, z_j, z_k) \in \mathcal{T} \times \mathcal{T} \times \mathcal{T} : (z_i, z_j) \in \mathcal{S}, \ell_i \neq \ell_k\}.\end{aligned}$$

The distance is learned using the following convex program:

$$\begin{aligned}\min_{\mathbf{M} \in \mathbb{S}_+^d} \quad & \sum_{(z_i, z_j) \in \mathcal{S}} d^2_{\mathbf{M}}(\mathbf{x_i}, \mathbf{x_j})\\ \text{s.t.} \quad & d^2_{\mathbf{M}}(\mathbf{x_i}, \mathbf{x_k}) - d^2_{\mathbf{M}}(\mathbf{x_i}, \mathbf{x_j}) \geq 1 \quad \forall (z_i, z_j, z_k) \in \mathcal{R}.\end{aligned} \tag{3.4}$$

Slack variables are added to get soft constraints. The authors developed a special-purpose solver (based on subgradient descent and careful book-keeping) that is able to deal with billions of constraints. In practice, LMNN is one of the best performing methods, although it is sometimes prone to overfitting due to the absence of regularization, as we will see in Chapter 6. Note that Park et al. (2011) developed an alternative algorithm for solving (3.4) based on column generation while Do et al. (2012) highlighted a relation between LMNN and Support Vector Machines.

3.2.1.3 Information-Theoretic Approaches

ITML (Davis et al.) Information-Theoretical Metric Learning (ITML), proposed by Davis et al. (2007), is an important work because it introduces LogDet divergence regularization that will later be used in several other Mahalanobis distance learning methods (e.g., Jain et al., 2008; Qi et al., 2009). This Bregman divergence on PSD matrices is defined as:

$$D_{ld}(\mathbf{M}, \mathbf{M_0}) = \text{trace}(\mathbf{M}\mathbf{M_0}^{-1}) - \log\det(\mathbf{M}\mathbf{M_0}^{-1}) - d,$$

where d is the dimension of the input space and $\mathbf{M_0}$ is some PSD matrix we want to remain close to. In practice, $\mathbf{M_0}$ is often set to $\mathbf{I}$ (the identity matrix) and thus the regularization aims at keeping the learned distance close to the Euclidean distance. The key feature of the LogDet divergence is that it is finite if and only if $\mathbf{M}$ is PSD. Therefore, minimizing $D_{ld}(\mathbf{M}, \mathbf{M_0})$ provides an automatic and cheap way of preserving the positive semi-definiteness of $\mathbf{M}$. The LogDet divergence is also rank-preserving: if the initial matrix $\mathbf{M_0}$ has rank r, the learned matrix will also have rank r.

ITML is formulated as follows:

$$\begin{aligned}\min_{\mathbf{M} \in \mathbb{S}_+^d} \quad & D_{ld}(\mathbf{M}, \mathbf{M_0})\\ \text{s.t.} \quad & d^2_{\mathbf{M}}(\mathbf{x_i}, \mathbf{x_j}) \leq u \quad \forall (z_i, z_j) \in \mathcal{S}\\ & d^2_{\mathbf{M}}(\mathbf{x_i}, \mathbf{x_j}) \geq v \quad \forall (z_i, z_j) \in \mathcal{D},\end{aligned} \tag{3.5}$$

where $u, v \in \mathbb{R}$ are threshold parameters (as usual, slack variables are added to get soft constraints). ITML thus aims at satisfying the similarity and dissimilarity constraints while staying as close as possible to the Euclidean distance (if $\mathbf{M_0} = \mathbf{I}$). More precisely, the information-theoretic interpretation behind minimizing $D_{ld}(\mathbf{M}, \mathbf{M_0})$ is that it is equivalent to minimizing the KL divergence between two multivariate Gaussian distributions parameterized by $\mathbf{M}$ and $\mathbf{M_0}$. The algorithm proposed to solve (3.5) is efficient, converges to the global minimum and the resulting distance performs well in practice. A limitation of ITML is that $\mathbf{M_0}$, that must be picked by hand, can have an important influence on the quality of the learned distance.

SDML (Qi et al.) With Sparse Distance Metric Learning (SDML), Qi et al. (2009) specifically deal with the case of high-dimensional data together with few training samples, i.e., $n \ll d$. To avoid overfitting, they use a double regularization: the LogDet divergence (using $\mathbf{M_0} = \mathbf{I}$ or $\mathbf{M_0} = \mathbf{\Sigma}^{-1}$) and L_1-regularization on the off-diagonal elements of $\mathbf{M}$. The justification for using this L_1-regularization is two-fold: (i) a practical one is that in high-dimensional spaces, the off-diagonal elements of $\mathbf{\Sigma}^{-1}$ are often very small, and (ii) a theoretical one suggested by a consistency result from a previous work in covariance matrix estimation that applies to SDML. They use a fast algorithm based on block-coordinate descent (the optimization is done over each row of $\mathbf{M^{-1}}$) and obtain very good performance for the specific case $n \ll d$.

3.2.1.4 Online Approaches

In online learning (Littlestone, 1988), the algorithm receives training instances one at a time and updates at each step the current hypothesis. Although the performance of online algorithms is typically inferior to batch algorithms, they are very useful to tackle large-scale problems that batch methods fail to address due to complexity and memory issues. Online learning methods often come with guarantees in the form of regret bounds, stating that the accumulated loss suffered along the way is not much worse than that of the best hypothesis chosen in hindsight.[1] However these results assume that the training pairs/triplets are generated i.i.d. (which is hardly the case in metric learning, as we will discuss later) and do not say anything about the generalization to unseen data.

POLA (Shalev-Shwartz et al.) POLA (Shalev-Shwartz et al., 2004) is the first online Mahalanobis distance learning approach and learns the matrix $\mathbf{M}$ as well as a threshold $b \geq 1$. At each step, when receiving the pair $(\mathbf{z_i}, \mathbf{z_j})$, POLA performs two successive orthogonal projections:

[1] A regret bound has the following general form: $\sum_{t=1}^{T} \ell(h, z_t) - \sum_{t=1}^{T} \ell(h^*, z_t) \leq O(T)$, where T is the number of steps and h^* is the best batch hypothesis.

1. Projection of the current solution $(\mathbf{M^{i-1}}, b^{i-1})$ onto $C_1 = \{(\mathbf{M}, b) \in \mathbb{R}^{d^2+1} : \mathbf{M}, b = [y_i y_j (d^2_{\mathbf{M}}(\mathbf{x_i}, \mathbf{x_j}) - b) + 1]_+ = 0\}$, which is done efficiently (closed-form solution). The constraint basically requires that the distance between two instances of same (resp. different) labels be below (resp. above) the threshold b with a margin 1. We get an intermediate solution $(\mathbf{M^{i-\frac{1}{2}}}, b^{i-\frac{1}{2}})$ that satisfies this constraint while staying as close as possible to the previous solution.

2. Projection of $(\mathbf{M^{i-\frac{1}{2}}}, b^{i-\frac{1}{2}})$ onto $C_2 = \{(\mathbf{M}, b) \in \mathbb{R}^{d^2+1} : \mathbf{M} \in \mathbb{S}^d_+, b \geq 1\}$, which is done rather efficiently (in the worst case, only needs to compute the minimal eigenvalue). This projects the matrix back onto the PSD cone. We thus get a new solution $(\mathbf{M^i}, b^i)$ that yields a valid Mahalanobis distance.

A regret bound for the algorithm is provided. However, POLA relies on the unrealistic assumption that there exists $(\mathbf{M}^*, b^*)$ such that $[y_i y_j (d^2_{\mathbf{M}^*}(\mathbf{x_i}, \mathbf{x_j}) - b^*) + 1]_+ = 0$ for all training pairs (i.e., there exists a matrix and a threshold value that perfectly separate them with margin 1), and is not competitive in practice.

LEGO (Jain et al.) LEGO, developed by Jain et al. (2008), is an improved version of POLA based on LogDet divergence regularization. It features tighter regret bounds, more efficient updates and better practical performance.

ITML (David et al.) ITML, presented in Section 3.2.1.3, also has an online version with bounded regret. At each step, the algorithm minimizes a tradeoff between LogDet regularization with respect to the previous matrix and a square loss. The resulting distance generally performs slightly worse than the batch version but the algorithm can be faster.

MDML (Kunapuli & Shavlik) The work of Kunapuli & Shavlik (2012) is an attempt of proposing a general framework for online Mahalanobis distance learning. It is based on composite mirror descent (Duchi et al., 2010), which allows online optimization of many regularized problems. It can accommodate a large class of loss functions and regularizers for which efficient updates are derived, and the algorithm comes with a regret bound. In the experiments, they focus on trace norm regularization, which is the best convex relaxation of the rank and thus induces low-rank matrices. In practice, the approach has performance comparable to LMNN and ITML, is fast and sometimes induces low-rank solutions, but surprisingly the algorithm was not evaluated on large-scale datasets.

3.2.1.5 Metric Learning with Generalization Guarantees

As in the classic supervised learning setting (where training data consist of individual labeled instances), generalization guarantees may be derived for supervised metric learning (where training data consist of pairs or triplets). Indeed, most of supervised metric learning methods can be seen as minimizing a (regularized) loss function ℓ based on the training pairs/triplets. In this context, the pair-based true risk can be defined as

$$R^{\ell}(d^2_{\mathbf{M}}) = \mathbb{E}_{z,z' \sim P}\left[\ell(d^2_{\mathbf{M}}, z, z')\right],$$

the pair-based empirical risk as

$$R^{\ell}_{\mathcal{S},\mathcal{D}}(d^2_{\mathbf{M}}) = \frac{1}{|\mathcal{S}| + |\mathcal{D}|} \sum_{(z_i,z_j) \in \mathcal{S} \cup \mathcal{D}} \ell(d^2_{\mathbf{M}}, z_i, z_j),$$

and likewise for the triplet-based setting.

However, although individual training instances are assumed to be drawn i.i.d. from P, one cannot make the same assumption regarding the pairs or triplets themselves since they are built from the training sample. For this reason, establishing generalization guarantees for the learned metric is challenging and has so far received very little attention. To the best of our knowledge, only two approaches have tried to address this question explicitly.

Jin et al. In their paper, Jin et al. (2009) study the following general Mahalanobis distance learning formulation:

$$\min_{\mathbf{M} \succeq 0} \quad \frac{1}{n^2} \sum_{(z_i,z_j) \in \mathcal{T} \times \mathcal{T}} \ell(d^2_{\mathbf{M}}, z_i, z_j) \quad + \quad C\|\mathbf{M}\|_{\mathcal{F}}, \tag{3.6}$$

where $C > 0$ is the regularization parameter. The loss function ℓ is assumed to be of the form

$$\ell(d^2_{\mathbf{M}}, z_i, z_j) = g(y_i y_j [1 - d^2_{\mathbf{M}}(\mathbf{x_i}, \mathbf{x_j})]),$$

where g is convex and Lipschitz continuous.

Relying on a definition of uniform stability adapted to the case of distance learning (where training data is made of pairs), they show that one can derive generalization bounds for the learned distance. Unfortunately, their framework is limited to Frobenius norm regularization: in particular, since it is based on uniform stability, it cannot accommodate sparsity-inducing regularizers. Note that they also propose an online algorithm that is efficient and competitive in practice.

The work of Jin et al. is related to the contributions of this thesis in two ways. First, in Chapter 5, we make use of the same uniform stability arguments to derive learning

guarantees for a learned edit similarity function, but we go a step further by deriving guarantees in terms of the error of the classifier built from this similarity. Second, in Chapter 7, we propose an alternative framework for deriving learning guarantees for metric learning based on algorithmic robustness, and we show that this framework can tackle a wider variety of problems.

Bian & Tao The work of Bian & Tao (2011; 2012) is another attempt of developing metric learning algorithms with generalization guarantees. They consider a class of loss functions similar to that of Jin et al. (2009):

$$\ell(d^2_{\mathbf{M}}, z_i, z_j) = g(y_{ij}[c - d^2_{\mathbf{M}}(\mathbf{x_i}, \mathbf{x_j})]),$$

where $c > 0$ is a decision threshold variable and g is convex and Lipschitz continuous.

The formulation they study is the following:

$$\min_{(\mathbf{M},c)\in\mathcal{Q}} \quad \frac{1}{n^2} \sum_{(z_i,z_j)\in\mathcal{T}\times\mathcal{T}} \ell(d^2_{\mathbf{M}}, z_i, z_j), \tag{3.7}$$

where $\mathcal{Q} = \{(\mathbf{M}, c) : 0 \preceq \mathbf{M} \preceq \alpha\mathbf{I}, 0 \leq c \leq \alpha\}$, with α a positive constant. This ensures that the learned metric and decision threshold are bounded.

They use a statistical analysis to derive risk bounds as well as consistency bounds (the learned distance asymptotically converges to the optimal distance). However, they rely on strong assumptions on the distribution of the examples and cannot accommodate any regularization.

3.2.1.6 Other approaches

In this section, we describe a few approaches that are outside the scope of the previous categories.

Rosales & Fung The method of Rosales & Fung (2006) aims at learning matrices with entire columns/rows set to zero, thus making $\mathbf{M}$ low-rank. For this purpose, they use L_1 norm regularization and, restricting their framework to diagonal dominant matrices, they are able to formulate the problem as a linear program that can be solved efficiently. However, L_1 norm regularization favors sparsity at the entry level only, not specifically at the row/column level, even though in practice the learned matrix is sometimes low-rank. Furthermore, the approach is less general than Mahalanobis distances due to the restriction to diagonal dominant matrices.

SML (Ying et al.) SML (Ying et al., 2009) is a Mahalanobis distance learning approach that regularizes $\mathbf{M}$ with the $L_{2,1}$ norm, which tends to zero out entire rows of $\mathbf{M}$ (as opposed to the L_1 norm used in the previous method). They essentially want to solve the following problem:

$$\begin{aligned} \min_{\mathbf{M}\in\mathbb{S}_+^d} \quad & \|\mathbf{M}\|_{2,1} \\ \text{s.t.} \quad & d^2_{\mathbf{M}}(\mathbf{x_i},\mathbf{x_k}) - d^2_{\mathbf{M}}(\mathbf{x_i},\mathbf{x_j}) \geq 1 \quad \forall (z_i,z_j,z_k)\in\mathcal{R}, \end{aligned}$$

where slack variables are added to get soft constraints. However, $L_{2,1}$ norm regularization is typically difficult to optimize. Using smoothing techniques the authors manage to derive an algorithm that scales in $O(d^3)$ per iteration. The method performs well in practice while inducing a lower-dimensional projection space than full-rank methods and the method of Rosales & Fung (2006). However, it cannot be applied to high-dimensional problems due to the complexity of the algorithm.

BoostMetric (Shen et al.) BoostMetric (Shen et al., 2009, 2012) adapts to Mahalanobis distance learning the ideas of boosting, where a good hypothesis is obtained through a weighted combination of so-called "weak learners" (Schapire & Freund, 2012, see the recent book on this matter by). The method is based on the property that any PSD matrix can be decomposed into a positive linear combination of trace-one rank-one matrices. This kind of matrices is thus used as weak learner and the authors adapt the popular boosting algorithm Adaboost (Freund & Schapire, 1995) to this setting. The resulting algorithm is quite efficient since it does not require full eigenvalue decomposition but only the computation of the largest eigenvalue. In practice, BoostMetric achieves competitive performance but hardly scales to large-scale or high-dimensional datasets.

DML (Ying et al.) The work of Ying & Li (2012) revisit MMC, the original approach of Xing et al. (2002), by investigating the following formulation, called DML-eig:

$$\begin{aligned} \max_{\mathbf{M}\in\mathbb{S}_+^d} \quad & \min_{(z_i,z_j)\in\mathcal{D}} d^2_{\mathbf{M}}(\mathbf{x_i},\mathbf{x_j}) \\ \text{s.t.} \quad & \sum_{(z_i,z_j)\in\mathcal{S}} d^2_{\mathbf{M}}(\mathbf{x_i},\mathbf{x_j}) \leq 1. \end{aligned} \tag{3.8}$$

The slight difference is that DML-eig (3.8) maximizes the *minimum* (square) distance between negative pairs while MMC (3.1) maximizes the sum of distances. Ying & Li avoid the costly full eigen-decomposition used by Xing et al. by showing that (3.8) can be cast as a well-known eigenvalue optimization problem called "minimizing the maximal eigenvalue of a symmetric matrix". They further show that it can be solved efficiently using a first-order algorithm that only requires the computation of the largest eigenvalue at each iteration, and that LMNN can also be cast as a similar problem.

Experiments show competitive results and low computational complexity, although it might be subject to overfitting due to the absence of regularization.

Cao et al. (2012) generalize (3.8) by studying the following formulation, called DML-p:

$$\begin{aligned} \max_{\mathbf{M}\in\mathbb{S}_+^d} \quad & \left(\frac{1}{|\mathcal{D}|}\sum_{(z_i,z_j)\in\mathcal{D}}[d_{\mathbf{M}}(\mathbf{x_i},\mathbf{x_j})]^{2p}\right)^{1/p} \\ \text{s.t.} \quad & \sum_{(z_i,z_j)\in\mathcal{S}} d^2_{\mathbf{M}}(\mathbf{x_i},\mathbf{x_j}) \leq 1. \end{aligned} \tag{3.9}$$

They show that for $p \in (-\infty, 1)$, (3.9) is convex and can be solved efficiently in an analogous manner as DML-eig. For $p = 0.5$ we recover MMC (3.1) and for $p \to -\infty$ we recover DML-eig (3.8). Experiments show that tuning p can lead to better performance than MMC or DML-eig.

LNML (Wang et al.) The idea of LNML (Wang et al., 2012) is to enhance metric learning methods by also learning the neighborhood (i.e., the pairs or triplets) according to which the metric is optimized. They propose an iterative approach that alternates between a neighbor assignment step (where the current metric is used to determine the neighbors according to some quality measure) and a metric learning step (where the metric is optimized with respect to the current neighborhood). Experiments are conducted on MCML and LMNN and show that more accurate metrics can be learned using their framework. Of course, this is achieved at the expense of higher computational complexity, since the metric learning algorithms must be run several times (5-10 times in their experiments).

3.2.2 Similarity Learning

Although most of the work in metric learning has focused on the Mahalanobis distance, learning similarity functions has also attracted some interest, motivated by the perspective of more scalable algorithms due to the absence of PSD constraint.

SiLA (Qamar et al.) SiLA (Qamar et al., 2008) is an approach for learning similarity functions of the following form:

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

where $\mathbf{M} \in \mathbb{R}^{d\times d}$ and $N(\mathbf{x},\mathbf{x}')$ is a normalization term which depends on $\mathbf{x}$ and $\mathbf{x}'$. This similarity function can be seen as a generalization of the cosine and the bilinear similarities. The authors build on the same idea of "target neighbors" that was introduced in LMNN, but optimize the similarity in an online manner with an algorithm based on voted perceptron. At each step, the algorithm goes through the training set, updating

the matrix when an example does not satisfy a criterion of separation. The authors present theoretical results that follow from the voted perceptron theory in the form of regret bounds for the separable and nonseparable cases. SiLA is compared to Mahalanobis metric learning approaches on three datasets. It seems to perform fine but has a rather slow convergence rate and may suffer of its lack of regularization. In subsequent work, Qamar & Gaussier (2012) study the relationship between SiLA and RELIEF, an online feature reweighting algorithm.

gCosLA (Qamar & Gaussier) gCosLA (Qamar & Gaussier, 2009) learns generalized cosine similarities of the form

$$\frac{\mathbf{x}^T\mathbf{M}\mathbf{x}'}{\sqrt{\mathbf{x}^T\mathbf{M}\mathbf{x}}\sqrt{\mathbf{x}'^T\mathbf{M}\mathbf{x}'}},$$

where $\mathbf{M} \in \mathbb{S}_+^d$. It corresponds to a cosine similarity in the projection space implied by $\mathbf{M}$. The algorithm itself, an online procedure, is very similar to that of POLA (presented in Section 3.2.1.4). Indeed, they essentially use the same loss function and also have a two-step approach: a projection onto the set of arbitrary matrices that achieve zero loss on the current example pair, followed by a projection back onto the PSD cone. The first projection is different from POLA (since the generalized cosine has a normalization factor that depends on $\mathbf{M}$) but the authors manage to derive a closed-form solution. The second projection is based on a full eigenvalue decomposition of $\mathbf{M}$, making the approach costly as dimensionality grows. A regret bound for the algorithm is provided and it is shown experimentally that gCosLA converges in fewer iterations than SiLA and is generally more accurate. Its performance seems competitive with LMNN and ITML.

OASIS (Chechik et al.) The similarity learning method OASIS (Chechik et al., 2009, 2010) learns a bilinear similarity $K_{\mathbf{M}}$ (see Section 2.3.2) for large-scale problems. Since $\mathbf{M} \in \mathbb{R}^{d\times d}$ is not required to be PSD, they can optimize the similarity in an online manner using a simple and efficient algorithm, which belongs to the family of Passive-Aggressive algorithms (Crammer et al., 2006). The initialization is $\mathbf{M} = \mathbf{I}$, then at each step t, the algorithm draws a triplet $(z_i, z_j, z_k) \in \mathcal{R}$ and solves the following convex problem:

$$\begin{aligned} \mathbf{M}^{\boldsymbol{t}} = \quad & \underset{\mathbf{M},\xi}{\arg\min} \quad \frac{1}{2}\|\mathbf{M} - \mathbf{M}^{\boldsymbol{t-1}}\|_{\mathcal{F}}^2 + C\xi \\ & \text{s.t.} \quad 1 - d_{\mathbf{M}}^2(\mathbf{x_i}, \mathbf{x_j}) + d_{\mathbf{M}}^2(\mathbf{x_i}, \mathbf{x_k}) \leq \xi \\ & \qquad \xi \geq 0, \end{aligned} \tag{3.10}$$

where C is the trade-off parameter between minimizing the loss and staying close from the matrix obtained at the previous step, and ξ is a slack variable. Clearly, if $1 - d_{\mathbf{M}}^2(\mathbf{x_i}, \mathbf{x_j}) + d_{\mathbf{M}}^2(\mathbf{x_i}, \mathbf{x_k} \leq 0$, then $\mathbf{M}^{\boldsymbol{t}} = \mathbf{M}^{\boldsymbol{t-1}}$ is the solution of (3.10). Otherwise, the solution is obtained from a simple closed-form update. In practice, OASIS achieves

competitive results on medium-scale problems and unlike most other methods, is scalable to problems with millions of training instances. However, it cannot incorporate complex regularizers and does not have generalization guarantees.

Note that the same authors derived two more algorithms for learning bilinear similarities as applications of more general frameworks. The first one is based on online learning in the manifold of low-rank matrices (Shalit et al., 2010, 2012) and the second one on adaptive regularization of weight matrices (Crammer & Chechik, 2012).

3.2.3 Nonlinear Metric Learning

We have seen that the work in supervised metric learning from feature vectors has focused on linear metrics because they are more convenient to optimize (in particular, it is easier to derive convex formulations with the guarantee of finding the global optimum) and less prone to overfitting. However, a drawback of linear metric learning is that it will fail to capture nonlinear patterns in the data.

An example of nonlinear metric learning is kernel learning, but existing approaches are very expensive and/or subject to local minima (see for instance Ong et al., 2002, 2005; Xu et al., 2012b), cannot be applied to unseen data (Lanckriet et al., 2002, 2004; Tsuda et al., 2005; Kulis et al., 2006, 2009) or limited to learning a combination of existing kernels such as in Multiple Kernel Learning (see Gönen & Alpaydn, 2011, for a recent survey).

So far, the most satisfactory solution to the problem of nonlinear metric learning is probably the kernelization of linear metric learning methods, in the spirit of what is done in SVM, i.e., learn a linear metric in the nonlinear feature space induced by a kernel function and thereby combine the best of both worlds. Some metric learning approaches have been shown to be kernelizable (for instance Schultz & Joachims, 2003; Shalev-Shwartz et al., 2004; Davis et al., 2007) using specific arguments, but in general kernelizing a particular metric algorithm is not trivial: a new formulation of the problem has to be derived, where interface to the data is limited to inner products, and sometimes a different implementation is necessary. Moreover, when kernelization is possible, one must learn a $n_{\mathcal{T}} \times n_{\mathcal{T}}$ matrix. As $n_{\mathcal{T}}$ gets large, the problem becomes intractable unless dimensionality reduction is applied. Recently though, several authors (Chatpatanasiri et al., 2010; Zhang et al., 2010) have proposed general kernelization methods based on Kernel Principal Component Analysis (Schölkopf et al., 1998). They can be used to kernelize nearly any metric learning algorithm and perform dimensionality reduction simultaneously in a very simple manner, referred to as the "KPCA trick". Since our bilinear similarity learning approach introduced in Chapter 6 is kernelized using this trick, we postpone the details to Section 6.2.2.

Note that kernelizing a metric learning algorithm may drastically improve the quality of the learned metric on highly nonlinear problems, but may also favor overfitting (because local pair or triplet-based constraints become much easier to satisfy in a nonlinear, high-dimensional kernel space), leading to poor generalization ability.

3.2.4 Approaches for Other Settings

In this review, we discussed metric learning approaches for the general supervised learning setting. Note that there also exist methods for the semi-supervised setting (Zha et al., 2009; Baghshah & Shouraki, 2009; Liu et al., 2010; Dai et al., 2012), domain adaptation (Cao et al., 2011; Geng et al., 2011; Kulis et al., 2011) and multi-task/view learning (Parameswaran & Weinberger, 2010; Wang et al., 2011; Yang et al., 2012). There also exists specific literature on metric learning for computer vision tasks such as object recognition (Frome et al., 2007; Verma et al., 2012), face recognition (Guillaumin et al., 2009) or tracking (Li et al., 2012).

3.3 Metric Learning from Structured data

As pointed out earlier, metrics have a special importance in the context of structured data: they can be used as a proxy to access data without having to manipulate these complex objects. As a consequence, given an appropriate structured metric, one can use k-NN, SVM, K-Means or any other metric-based algorithm as if the data consisted of feature vectors.

Unfortunately, for the same reasons, metric learning from structured data is challenging because most of structured metrics are combinatorial by nature, which explains why it has received less attention than metric learning from feature vectors. Most of the available literature on the matter focuses on learning metrics based on the edit distance. Clearly, for the edit distance to be meaningful, one needs costs that reflect the reality of the considered task. To take a simple example, in typographical error correction, the probability that a user hits the Q key instead of W on a QWERTY keyboard is much higher than the probability that he hits Q instead of Y. For some applications, such as protein alignment or handwritten digit recognition, well-tailored cost matrices may be available (Dayhoff et al., 1978; Henikoff & Henikoff, 1992; Micó & Oncina, 1998). Otherwise, there is a need for automatically learning a nonnegative $(|\Sigma|+1)\times(|\Sigma|+1)$ cost matrix $\mathbf{C}$ for the task at hand.

What makes the cost matrix difficult to optimize is the fact that the edit distance is based on an optimal script which depends on the edit costs themselves. Most general-purpose approaches get round this problem by considering a stochastic variant of the edit distance, where the cost matrix defines a probability distribution over the edit

operations. One can then define an edit similarity equal to the posterior probability $p_e(\mathsf{x}'|\mathsf{x})$ that an input string x is turned into an output string x'. This corresponds to summing over all possible edit scripts that turn x into x' instead of only considering the optimal script. Such a stochastic edit process can be represented as a probabilistic model and one can estimate the parameters (i.e., the cost matrix) of the model that maximize the expected log-likelihood of positive pairs. This is done via an iterative Expectation-Maximization (EM) algorithm (Dempster et al., 1977), a procedure that alternates between two steps: an Expectation step (which essentially computes the function of the expected log-likelihood of the pairs with respect to the current parameters of the model) and a Maximization step (computing the updated edit costs that maximize this expected log-likelihood). Note that unlike the classic edit distance, the obtained edit similarity does not usually satisfy the properties of a distance (in fact, it is often not symmetric).

In the following, we review methods for learning string edit metrics (Section 3.3.1) and tree edit metrics (Section 3.3.2).

3.3.1 String Edit Metric Learning

Generative models The first method for learning a string edit metric was proposed by Ristad & Yianilos (1998). They use a memoryless stochastic transducer which models the joint probability of a pair $p_e(\mathsf{x}, \mathsf{x}')$ from which $p_e(\mathsf{x}'|\mathsf{x})$ can be estimated. Parameter estimation is performed with EM and the learned edit probability is applied to the problem of learning word pronunciation in conversational speech. Bilenko & Mooney (2003) extended this approach to the Needleman-Wunsch Score with affine gap penalty and applied it to duplicate detection. To deal with the tendency of Maximum Likelihood estimators to overfit when the number of parameters is large (in this case, when the alphabet size is large), Takasu (2009) proposes a Bayesian parameter estimation of pair-HMM providing a way to smooth the estimation. Experiments are conducted on approximate text searching in a digital library of Japanese and English documents.

Discriminative models The work of Oncina & Sebban (2006) describes three levels of bias induced by the use of generative models: (i) dependence between edit operations, (ii) dependence between the costs and the prior distribution of strings $p_e(\mathsf{x})$, and (iii) the fact that to obtain the posterior probability one must divide by the empirical estimate of $p_e(\mathsf{x})$. These biases are highlighted by empirical experiments conducted with the method of Ristad & Yianilos (1998). To address these limitations, they propose the use of a conditional transducer that directly models the posterior probability $p_e(\mathsf{x}'|\mathsf{x})$ that an input string x is turned into an output string x' using edit operations. Parameter estimation is also done with EM and the paper features an application to handwritten digit recognition, where digits are represented as sequences of Freeman codes (Freeman,

1974). In order to allow the use of negative pairs, McCallum et al. (2005) consider another discriminative model, conditional random fields, that can deal with positive and negative pairs in specific states, still using EM for parameter estimation.

Methods based on gradient descent The use of EM has two main drawbacks: (i) it may converge to a local optimum, and (ii) parameter estimation and distance calculations must be done at each iteration, which can be very costly if the size of the alphabet and/or the length of the strings are large.

Saigo et al. (2006) manage to avoid the need for an iterative procedure like EM in the context of detecting remote homology in protein sequences. They learn the parameters of the Smith-Waterman score which is plugged in their local alignment kernel (Saigo et al., 2004). Unlike the Smith-Waterman score, the local alignment kernel, which is based on the sum over all possible alignments, is differentiable and can be optimized by a gradient descent procedure. The objective function that they optimize is meant to favor the discrimination between positive and negative examples, but this is done by only using positive pairs of distant homologs. The approach has two additional drawbacks: (i) the objective function is nonconvex and it thus subject to local minima, and (ii) the kernel's validity is not guaranteed in general and is subject to the value of a parameter that must be tuned. Therefore, the authors use this learned function as a similarity measure and not as a kernel.

3.3.2 Tree Edit Metric Learning

Bernard et al. Extending the work of Ristad & Yianilos (1998) and Oncina & Sebban (2006) on string edit similarity learning, Bernard et al. (2006, 2008) propose both a generative and a discriminative model for learning tree edit costs. They rely on the tree edit distance by Selkow (1977) — which is cheaper to compute than that of Zhang & Shasha (1989) — and adapt the updates of EM to this case. An application to handwritten digit recognition is proposed, where digits are represented by trees of Freeman codes.

Boyer et al. The work of Boyer et al. (2007) tackles the more complex variant of the tree edit distance (Zhang & Shasha, 1989), which allows the insertion and deletion of single nodes instead of entire subtrees only. Parameter estimation in the generative model is also based on EM, and the usefulness of the approach is illustrated on an image recognition task.

Neuhaus & Bunke In their paper, Neuhaus & Bunke (2007) learn a (more general) graph edit similarity, where each edit operation is modeled by a Gaussian mixture

density. Parameter estimation is done using an EM-like algorithm. Unfortunately, the approach is intractable: the complexity of the EM procedure is exponential in the number of nodes (and so is the computation of the distance).

Dalvi et al. The work of Dalvi et al. (2009) points out a limitation of the approach of Bernard et al. (2006, 2008): they model a distribution over tree edit scripts rather than over the trees themselves, and unlike the case of strings, there is no bijection between the edit scripts and the trees. Recovering the correct conditional probability with respect to trees requires a careful and costly procedure. They propose a more complex conditional transducer that models the conditional probability over trees and use EM for parameter estimation. They apply their method to the problem of creating robust wrappers for webpages.

Emms The work of Emms (2012) points out a theoretical limitation of the approach of Boyer et al. (2007): the authors use a factorization that turns out to be incorrect in some cases. Emms shows that a correct factorization exists when only considering the edit script of highest probability instead of all possible scripts, and derives the corresponding EM updates. An obvious drawback is that the output of the model is not the probability $p_e(\mathsf{x}'|\mathsf{x})$. Moreover, experiments on a question answering task highlight that the approach is prone to overfitting, and requires smoothing and other heuristics (such as a final step of zeroing-out the diagonal of the cost matrix).

3.4 Conclusion

In this chapter, we reviewed a large body of work in supervised metric learning. Table 3.1 and Table 3.2 summarize the main features of the studied approaches for feature vectors and structured data respectively.

This review raises three observations:

1. Research efforts on metric learning from feature vectors have been mainly oriented towards deriving tractable formulations and algorithms. Boosted by some advances in batch and online numerical optimization, these efforts have been successful: recent methods are scalable and can even accommodate complex regularizers in an efficient way. However, there is an obvious lack of theoretical understanding of metric learning. First, few frameworks capable of establishing the consistency of the learned metric on unseen data have been proposed, and existing ones lack generality. Second, using a learned metric often improves the empirical performance of metric-based algorithms, but this has never been studied from a theoretical

Method	Convex	Scalable	Competitive	Reg.	Low-rank	Online	Gen.
MMC	✓	✗	✗	✗	✗	✗	✗
Schultz & Joachims	✓	✓	✗	✓	✗	✗	✗
NCA	✗	✓	✗	✗	✓	✗	✗
MCML	✓	✗	✗	✗	✗	✗	✗
LMNN	✓	✓✓	✓	✗	✗	✗	✗
ITML	✓	✓✓	✓	✓	✓	✓	✗
SDML	✓	✓✓	✓	✓	✗	✗	✗
POLA	✓	✓	✗	✗	✗	✓	✗
LEGO	✓	✓✓	✓	✓	✓	✓	✗
MDML	✓	✓✓	✓	✓	✓	✓	✗
Jin et al.	✓	✓✓	✓	✓	✗	✓	✓
Bian & Tao	✓	✓	✓	✗	✗	✗	✓
Rosales & Fung	✓	✓	✗	✓	✓	✗	✗
SML	✓	✗	✓	✓	✓	✗	✗
BoostMetric	✓	✓	✓	✓	✗	✗	✗
DML	✓	✓✓	✓	✗	✗	✗	✗
SiLA	—	✓	?	✗	✗	✓	✗
gCosLA	✓	✓	✓	✗	✗	✓	✗
OASIS	✓	✓✓✓	✓	✓	✗	✓	✗

TABLE 3.1: Summary of the main features of the reviewed approaches ("Reg." and "Gen." respectively stand for "Regularized" and "Generalization guarantees").

Method	Data	Model	Scripts	Opt.	Global sol.	Neg. pairs	Gen.
Ristad & Yianilos	Strings	Generative	All	EM	✗	✗	✗
Bilenko & Mooney	Strings	Generative	All	EM	✗	✗	✗
Takasu	Strings	Generative	All	EM	✗	✗	✗
Oncina & Sebban	Strings	Discriminative	All	EM	✗	✗	✗
McCallum et al.	Strings	Discriminative	All	EM	✗	✓	✗
Saigo et al.	Strings	—	All	GD	✗	✗	✗
Bernard et al.	Trees	Both	All	EM	✗	✗	✗
Boyer et al.	Trees	Generative	All	EM	✗	✗	✗
Neuhaus & Bunke	Graphs	Generative	All	EM	✗	✗	✗
Dalvi et al.	Trees	Discriminative	All	EM	✗	✗	✗
Emms	Trees	Discriminative	Optimal	EM	✗	✗	✗

TABLE 3.2: Summary of the main features of the reviewed approaches ("Opt.", "Global sol.", "Neg. pairs" and "Gen." respectively stand for "Optimization", "Global solution", "Negative pairs" and "Generalization guarantees").

standpoint. In particular, can we relate the empirical risk of the learned metric to the true risk of the classifier that uses it?

2. There is a relatively small body of work on metric learning from structured data, presumably due to the higher complexity of the learning procedures. Almost all existing methods are based on probabilistic models: they are trained using an expensive iterative algorithm and cannot accommodate negative pairs. Furthermore, no approach is guaranteed to converge to the global optimum of the optimized quantity and again, there is a lack of theoretical study.

3. The use of learned metrics is typically restricted to algorithms based on local neighborhoods, in particular k-NN classifiers. Since the learned metrics are typically optimized over local constraints, it seems unclear whether they can be successfully

used in more global classifiers such as SVM and other linear separators, or if new metric learning algorithms should be designed for this global setting. Furthermore, building a PSD kernel from the learned metrics is often difficult, especially for structured data (e.g., string edit kernels).

The contributions of this thesis address these limitations. Part II is devoted to metric learning from structured data and consists of two main contributions. In Chapter 4, we introduce a new string kernel built from learned edit probabilities. Unlike other string edit kernels, it is guaranteed to be PSD and parameter-free. In Chapter 5, we propose a novel string and tree edit similarity learning method based on numerical optimization, that can handle positive and negative pairs and is guaranteed to converge to the optimal solution. We are able to derive a generalization bound for our method, and this bound can be related to the generalization error of a linear classifier built from the learned similarity. Part III is devoted to metric learning from feature vectors and consists of two contributions. In Chapter 6, we propose a bilinear similarity learning method tailored to linear classification. The similarity is not optimized over local pair or triplet-based constraints: it directly minimizes a global quantity that upper bounds the true risk of the linear classifier built from the learned similarity. Lastly, in Chapter 7, we adapt the notion of algorithmic robustness (Section 2.2.3) to the metric learning setting, which allows us to derive generalization guarantees for a large class of metric learning problems with various loss functions and regularizers.

PART II

Contributions in Metric Learning from Structured Data

CHAPTER 4
A String Kernel Based on Learned Edit Similarities

Chapter abstract With the success of kernel methods, there is a growing interest in designing powerful kernels between sequences. In this chapter, we propose a new string kernel based on edit probabilities learned with a conditional transducer. Unlike other string edit kernels, it is parameter-free and guaranteed to be valid since it corresponds to a dot product in an infinite-dimensional space. While the naive computation of the kernel involves an intractable sum over an infinite number of strings, we show that it can actually be computed exactly using the intersection of probabilistic automata and a matrix inversion. Experimental results on a handwritten character recognition task show that our new kernel outperforms state-of-the-art string kernels as well as standard and learned edit distance used in a k-NN classifier.

The material of this chapter is based on the following international publication:

Aurélien Bellet, Marc Bernard, Thierry Murgue, and Marc Sebban. Learning state machine-based string edit kernels. *Pattern Recognition (PR)*, 43(6):2330–2339, 2010.

4.1 Introduction

In recent years, with the emergence of kernel-based learning, a lot of research has gone into designing powerful kernels for structured data such as strings. A natural way of building string kernels consists in representing each sequence by a fixed-length feature vector. Many of the early string kernels, such as the spectrum, the subsequence or the mismatch kernels (presented in Section 2.3.3), belong to this family. They sometimes perform well, but they are not very flexible and imply a significant loss of structural information.

On the other hand, measures based on (or related to) the string edit distance can capture more structural distortions and are adaptable by nature, since they are based on a cost matrix that can be used to incorporate background knowledge on the problem of interest. When a domain expertise is not available, one may learn these costs automatically from data (we have reviewed these methods in Section 3.3). Unfortunately, their use is mostly restricted to k-NN classifiers since efforts to design string kernels from the edit distance

(the so-called edit kernels) have not been satisfactory: their validity (i.e., positive semi-definiteness) is subject to the value of a parameter (that must be tuned) and/or they suffer from the "diagonal dominance" problem (Li & Jiang, 2004; Cortes et al., 2004; Saigo et al., 2004; Neuhaus & Bunke, 2006). Another drawback of these approaches is that they use the standard version of the edit distance. Adapting them to make use of learned edit similarities (that are not proper distances and sometimes not even symmetric) is often not straightforward.

In this work, we propose a new string edit kernel that makes use of conditional edit probabilities learned in the form of probabilistic models. Our kernel belongs to the family of marginalized kernels (Tsuda et al., 2002; Kashima et al., 2003), is parameter-free and guaranteed to be PSD. It also has the unusual feature of being based on a sum over an infinite number of strings. This sum may seem intractable at first glance, but drawing our inspiration from rational kernels (Cortes et al., 2004), we show that it can be computed exactly by means of the intersection of two probabilistic automata and a matrix inversion. We conduct experiments on a handwritten digit recognition task that show that our kernel outperforms state-of-the-art string kernels, as well as k-NN classifiers based on standard and learned edit distance measures.

The rest of this chapter is organized as follows. Section 4.2 introduces our new string edit kernel. Section 4.3 is devoted to the computation of the kernel based on intersection of probabilistic automata and matrix inversion. Experimental results are presented in Section 4.4 and we conclude in Section 4.5.

4.2 A New Marginalized String Edit Kernel

Our new string edit kernel belongs to the family of marginalized kernels (Tsuda et al., 2002; Kashima et al., 2003). Let $p(\mathsf{x}, \mathsf{x}', v)$ be the probability of observing jointly a hidden variable $v \in \mathcal{V}$ and two observable strings $\mathsf{x}, \mathsf{x}' \in \Sigma^*$. The probability $p(\mathsf{x}, \mathsf{x}')$ can be obtained by marginalizing, *i.e.* summing over all variables $v \in \mathcal{V}$, the probability $p(\mathsf{x}, \mathsf{x}', v)$, such that:

$$p(\mathsf{x}, \mathsf{x}') = \sum_{v \in \mathcal{V}} p(\mathsf{x}, \mathsf{x}', v) = \sum_{v \in \mathcal{V}} p(\mathsf{x}, \mathsf{x}' | v) \cdot p(v).$$

A marginalized kernel computes this probability making the assumption that x and x' are conditionally independent given v, i.e.,

$$K(\mathsf{x}, \mathsf{x}') = \sum_{v \in \mathcal{V}} p(\mathsf{x}|v) \cdot p(\mathsf{x}'|v) \cdot p(v). \tag{4.1}$$

Note that the computation of this kernel is possible since it is assumed that $\mathcal{V}$ is a *finite set*. Let us now suppose that $p(v|\mathsf{x})$ is known instead of $p(\mathsf{x}|v)$. Then, as described in

(Tsuda et al., 2002), we can use the following marginalized kernel:

$$K(\mathsf{x}, \mathsf{x}') = \sum_{v \in \mathcal{V}} p(v|\mathsf{x}) \cdot p(v|\mathsf{x}') \cdot K_c(c, c'), \tag{4.2}$$

where $K_c(c, c')$ is the joint kernel depending on combined variables $c = (\mathsf{x}, v)$ and $c' = (\mathsf{x}', v)$. An interesting way to exploit the kernel in Equation 4.2 as a string edit kernel is to do the following:

- replace the finite set $\mathcal{V}$ of variables v by the *infinite* set of strings $\mathsf{s} \in \Sigma^*$,
- use $p(\mathsf{s}|\mathsf{x}) = p_e(\mathsf{s}|\mathsf{x})$, which is the conditional probability that a string x is turned into a string s through edit operations,
- and take $K_c(c, c')$ to be the constant kernel that returns 1 for all c, c'.

We then obtain the following new string edit kernel:

$$K_e(\mathsf{x}, \mathsf{x}') = \sum_{\mathsf{s} \in \Sigma^*} p_e(\mathsf{s}|\mathsf{x}) \cdot p_e(\mathsf{s}|\mathsf{x}'), \tag{4.3}$$

which is PSD as it corresponds to the inner product in the Hilbert space defined by the mapping $\phi(\mathsf{x}) = [p_e(\mathsf{s}|\mathsf{x})]_{\mathsf{s} \in \Sigma^*}$. Like the popular Gaussian kernel for feature vectors, K_e projects the data into an infinite-dimensional space. Intuitively, $K_e(\mathsf{x}, \mathsf{x}')$ is large when x and x' have a high probability to be turned into the same strings $\mathsf{s} \in \Sigma^*$ using edit operations.

We have already seen in Section 3.3.1 that there exist methods in the literature for learning $p_e(\mathsf{x}'|\mathsf{x})$ for all x, x'. However, our new kernel is intractable in its current form since it involves the computation of an infinite sum over Σ^*. In the next section, we present a way of computing this infinite sum exactly and in an efficient way.

4.3 Computing the Edit Kernel

While the original marginalized kernel (4.1) assumes that $\mathcal{V}$ is a finite set of variables (Tsuda et al., 2002), our string edit kernel includes an infinite sum over Σ^*. In this section, we show that (i) given two strings x and x', $p_e(\mathsf{s}|\mathsf{x})$ and $p_e(\mathsf{s}|\mathsf{x}')$ can be represented in the form of two probabilistic automata, (ii) the product $p_e(\mathsf{s}|\mathsf{x}) \cdot p_e(\mathsf{s}|\mathsf{x}')$ can be performed by intersecting the languages represented by those automata and (iii) the infinite sum over Σ^* can then be computed by algebraic methods.

4.3.1 Definitions and Notations

We first introduce some definitions and notations regarding probabilistic transducers.

Definition 4.1. A weighted finite-state transducer (WFT) is an 8-tuple $T = (\Sigma, \Delta, \mathcal{Q}, \mathcal{I}, \mathcal{F}, w, \tau, \rho)$ where Σ is the input alphabet, Δ the output alphabet, $\mathcal{Q}$ a finite set of states, $\mathcal{I} \subseteq \mathcal{Q}$ the set of initial states, $\mathcal{F} \subseteq \mathcal{Q}$ the set of final states, $w : \mathcal{Q} \times \mathcal{Q} \times (\Sigma \cup \{\$\}) \times (\Delta \cup \{\$\}) \to \mathbb{R}$ the transition weight function, $\tau : \mathcal{I} \to \mathbb{R}$ the initial weight function, and $\rho : \mathcal{F} \to \mathbb{R}$ the final weight function. For notational convenience, we denote $w(q_1, q_2, \mathsf{a}, \mathsf{b})$ by $w_{q_1 \to q_2}(\mathsf{a}, \mathsf{b})$ for any $q_1, q_2 \in \mathcal{Q}$, $\mathsf{a} \in \Sigma$ and $\mathsf{b} \in \Delta$.

Definition 4.2. A joint probabilistic finite-state transducer (jPFT) is a WFT $J = (\Sigma, \Delta, \mathcal{Q}, \mathcal{S}, \mathcal{F}, w, \tau, \rho)$ which defines a joint probability distribution over pairs of strings $\{(\mathsf{x}, \mathsf{x}') \in \Sigma^* \times \Delta^*\}$. A jPFT must satisfy the following four constraints:

1. The initial, final and transition weights have nonnegative values.
2. $\sum_{i \in \mathcal{S}} \tau(i) = 1$,
3. $\sum_{f \in \mathcal{F}} \rho(f) = 1$,
4. $\forall q_1 \in \mathcal{Q} : \displaystyle\sum_{q_2 \in \mathcal{Q}, \mathsf{a} \in \Sigma \cup \{\$\}, \mathsf{b} \in \Delta \cup \{\$\}} w_{q_1 \to q_2}(\mathsf{a}, \mathsf{b}) = 1.$

Definition 4.3. A conditional probabilistic finite-state transducer (cPFT) is a WFT $C = (\Sigma, \Delta, Q, S, F, w, \tau, \rho)$ which defines a conditional probability distribution over the output strings $\mathsf{x}' \in \Delta^*$ given an input string $\mathsf{x} \in \Sigma^*$. For $q_1, q_2 \in \mathcal{Q}$, $\mathsf{a} \in \Sigma$ and $\mathsf{b} \in \Delta$, we denote the transition $w_{q_1 \to q_2}(a, b)$ in the conditional form $w_{q_1 \to q_2}(b|a)$. A cPFT must satisfy the same first two constraints as those of a jPFT and the following third constraint (see Oncina & Sebban, 2006, for a proof):

$$\forall q_1 \in \mathcal{Q}, \forall \mathsf{a} \in \Sigma : \sum_{q_2 \in \mathcal{Q}, \mathsf{b} \in \Delta \cup \{\$\}} w_{q_1 \to q_2}(\mathsf{b}|\mathsf{a}) + w_{q_1 \to q_2}(\mathsf{b}|\$) = 1.$$

An example of memoryless cPFT (i.e., with only one state) is shown in Figure 4.1, where $\Sigma = \Delta = \{a, b\}$ and $\mathcal{Q}$ is composed of only one state labeled by 0. Initial states are designated by an inward arrow that has no source state, while final states are denoted by a double circle. In Figure 4.1, state 0 is both initial and final.

In the following, since our string edit kernel is based on conditional edit probabilities, we will assume that a cPFT has already been learned by one of the previously mentioned methods, for instance that of Oncina & Sebban (2006) that we describe in more details in Appendix A for the sake of completeness. Note that the cPFT is learned only once and is then used to compute our edit kernel for any pair of strings. If a generative model is used to learn the edit parameters (e.g., that of Ristad & Yianilos, 1998), the resulting jPFT can be renormalized into a cPFT a posteriori.

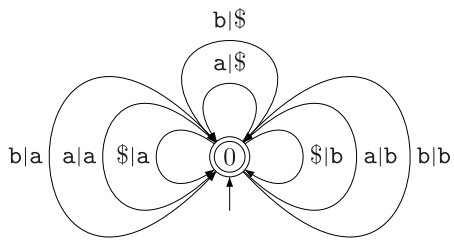


FIGURE 4.1: A memoryless cPFT that can be used to compute the edit conditional probability of any pair of strings. Edit probabilities assigned to each transition are not shown here for the sake of readability.

4.3.2 Modeling $p_e(\mathsf{s}|\mathsf{x})$ and $p_e(\mathsf{s}|\mathsf{x}')$ with Probabilistic Automata

Since our edit kernel $K_e(\mathsf{x}, \mathsf{x}')$ depends on two observable strings x and x', it is possible to represent the distributions $p_e(\mathsf{s}|\mathsf{x})$ and $p_e(\mathsf{s}|\mathsf{x}')$ in the form of probabilistic state machines, where only s is a hidden variable. Given a cPFT T modeling the edit probabilities and a string x, we can define a new cPFT driven by x, denoted by $T|\mathsf{x}$, that models $p_e(\mathsf{s}|\mathsf{x})$.

Definition 4.4. Let $T = (\Sigma, \Delta, \mathcal{Q}, \mathcal{S}, \mathcal{F}, w, \tau, \rho)$ be a cPFT that models $p_e(\mathsf{x}'|\mathsf{x})$, $\forall \mathsf{x}' \in \Delta^*, \forall \mathsf{x} \in \Sigma^*$. We define $T|\mathsf{x}$ as a cPFT that models $p_e(\mathsf{s}|\mathsf{x}), \forall \mathsf{s} \in \Delta^*$ and a specific observable $\mathsf{x} \in \Sigma^*$. $T|\mathsf{x} = (\Sigma, \Delta, \mathcal{Q}', \mathcal{S}', \mathcal{F}', w', \tau', \rho')$ with:

- $\mathcal{Q}' = \{[\mathsf{x}]_i\} \times \mathcal{Q}$ where $[\mathsf{x}]_i$ is the prefix of length i of x (note that $[\mathsf{x}]_0 = \$$). In other words, $\mathcal{Q}'$ is a finite set of states labeled by the current prefix of x and its corresponding state during its parsing in T.[1]
- $\mathcal{S}' = \{(\$, q)\}$ where $q \in \mathcal{S}$;
- $\forall q \in \mathcal{S}, \tau'((\$, q)) = \tau(q)$;
- $\mathcal{F}' = \{(\mathsf{x}, q)\}$ where $q \in \mathcal{F}$;
- $\forall q \in \mathcal{F}, \rho'((\mathsf{x}, q)) = \rho(q)$;
- the following two rules are used to define the transition weight function:
  - $\forall \mathsf{b} \in \Delta \cup \{\$\}, \forall q_1, q_2 \in \mathcal{Q}, w'_{([\mathsf{x}]_i, q_1) \to ([\mathsf{x}]_{i+1}, q_2)}(\mathsf{b}|\mathsf{x}_{\mathsf{i+1}}) = w_{q_1 \to q_2}(\mathsf{b}|\mathsf{x}_{\mathsf{i+1}})$,
  - $\forall \mathsf{b} \in \Delta, \forall q_1, q_2 \in \mathcal{Q}, w'_{([\mathsf{x}]_i, q_1) \to ([\mathsf{x}]_i, q_2)}(b|\$) = w_{q_1 \to q_2}(\mathsf{b}|\$)$.

As an example, given two strings $\mathsf{x} = \mathsf{a}$ and $\mathsf{x}' = \mathsf{ab}$, Figure 4.2 shows the cPFT $T|\mathsf{a}$ and $T|\mathsf{ab}$ constructed from the memoryless transducer T given in Figure 4.1. Roughly speaking, $T|\mathsf{a}$ and $T|\mathsf{ab}$ model the output languages that can be generated through edit operations from x and x' respectively. Therefore, from these state machines, we can generate output strings and compute the conditional edit probabilities $p_e(\mathsf{s}|\mathsf{x})$ and

[1] This specific notation is required to deal with nonmemoryless cPFT.

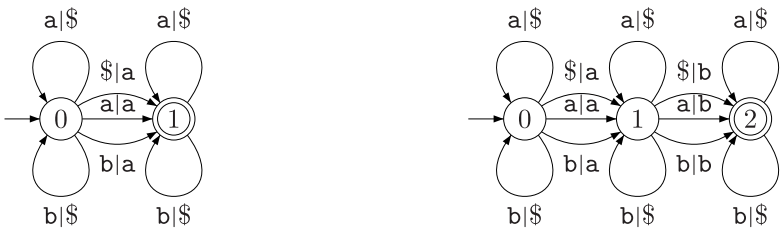


FIGURE 4.2: On the left: a cPFT $T|\mathsf{a}$ that models the output distribution conditionally to an input string $\mathsf{x} = \mathsf{a}$. For the sake of readability, state 0 stands for $(\$, 0)$, and state 1 for $(\mathsf{a}, 0)$. On the right: a cPFT $T|\mathsf{ab}$ that models the output distribution given $\mathsf{x}' = \mathsf{ab}$. Again, 0 stands for $(\$, 0)$, 1 for $(\mathsf{a}, 0)$, and 2 for $(\mathsf{ab}, 0)$.

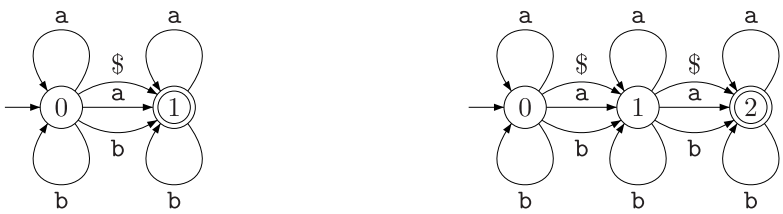


FIGURE 4.3: The cPFT $T|\mathsf{a}$ and $T|\mathsf{ab}$ of Figure 4.2 represented in the form of automata.

$p_e(\mathsf{s}|\mathsf{x}')$ for any string $\mathsf{s} \in \Delta^*$. Note that the cycles outgoing from each state model the possible insertions before and after reading an input symbol.

Since the construction of $T|\mathsf{x}$ and $T|\mathsf{x}'$ is driven by the parsing of x and x' in T, we can omit the input alphabet Σ. Therefore, a transducer $T|\mathsf{x} = (\Sigma, \Delta, \mathcal{Q}, \mathcal{S}, \mathcal{F}, w, \tau, \rho)$ can be reduced to a finite-state automaton $A|\mathsf{x} = (\Delta, \mathcal{Q}, \mathcal{S}, \mathcal{F}, w', \tau, \rho)$. The transitions of $A|\mathsf{x}$ are derived from w in the following way: $w'_{q_1 \to q_2}(\mathsf{b}) = w_{q_1 \to q_2}(\mathsf{b}|\mathsf{a}), \forall \mathsf{b} \in \Delta \cup \{\$\}, \forall \mathsf{a} \in \Sigma \cup \{\$\}, \forall q_1, q_2 \in \mathcal{Q}$. For example, Figure 4.3 shows the resulting automata deduced from the cPFT $T|\mathsf{a}$ and $T|\mathsf{ab}$ depicted in Figure 4.2.

4.3.3 Computing the Product $p_e(\mathsf{s}|\mathsf{x}) \cdot p_e(\mathsf{s}|\mathsf{x}')$

The next step for computing our kernel $K_e(\mathsf{x}, \mathsf{x}')$ is to compute the product $p_e(\mathsf{s}|\mathsf{x}) \cdot p_e(\mathsf{s}|\mathsf{x}')$. This can be performed by modeling the language that describes the intersection of the automata corresponding to $p_e(\mathsf{s}|\mathsf{x})$ and $p_e(\mathsf{s}|\mathsf{x}')$. This intersection can be obtained by performing a composition of transducers (Cortes et al., 2004). As mentioned by the authors, composition is a fundamental operation on weighted transducers that can be used to create complex weighted transducers from simpler ones. In this context, note that the intersection of two probabilistic automata (such as those of Figure 4.3) is a special case of composition where the input and output transition labels are identical. This intersection takes the form of a probabilistic automaton as defined below.

Definition 4.5. Let T be a cPFT modeling conditional edit probabilities. Let x and x' be two strings of Σ^*. Let $A|\mathsf{x} = (\Delta, \mathcal{Q}, \mathcal{S}, \mathcal{F}, w, \tau, \rho)$ and $A|\mathsf{x}' = (\Delta, \mathcal{Q}', \mathcal{S}', \mathcal{F}', w', \tau', \rho')$

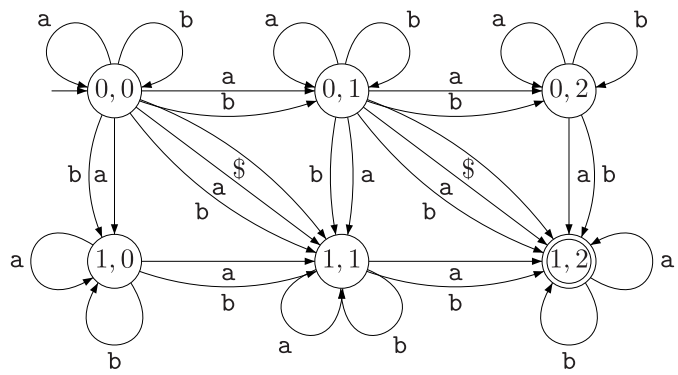


FIGURE 4.4: Automaton modeling the intersection of the automata of Figure 4.3.

be the automata deduced from T given the observable strings x and x'. We define the intersection of $A|\mathsf{x}$ and $A|\mathsf{x}'$ as the automaton $A|\mathsf{x},\mathsf{x}' = (\Delta, \mathcal{Q}^A, \mathcal{S}^A, \mathcal{F}^A, w^A, \tau^A, \rho^A)$ such that:

- $\mathcal{Q}^A = \mathcal{Q} \times \mathcal{Q}'$,
- $\mathcal{S}^A = \{(q, q')\}$ with $q \in \mathcal{S}$ and $q' \in \mathcal{S}'$,
- $\mathcal{F}^A = \{(q, q')\}$ with $q \in \mathcal{F}$ and $q' \in \mathcal{F}'$,
- $w^A_{(q_1,q_1')\to(q_2,q_2')}(\mathsf{b}) = w_{q_1\to q_2}(\mathsf{b}) \cdot w'_{q_1'\to q_2'}(\mathsf{b})$,
- $\tau^A((q, q')) = \tau(q) \cdot \tau(q')$,
- $\rho^A((q, q')) = \rho(q) \cdot \rho(q')$.

Figure 4.4 shows the intersection automaton of the two automata from Figure 4.3. Let us now describe how this intersection automaton can be used to compute the infinite sum over Σ^*.

4.3.4 Computing the Sum over Σ^*

To simplify the notations, let $p(\mathsf{s}) = p_e(\mathsf{s}|\mathsf{x}) \cdot p_e(\mathsf{s}|\mathsf{x}')$ be the probability that a string s is generated by an intersection automaton $A = \{\Sigma, \mathcal{Q}, \mathcal{S}, \mathcal{F}, w, \tau, \rho\}$ and $\Sigma = \{\mathsf{a}_1, \ldots, \mathsf{a}_{|\Sigma|}\}$ be the alphabet.

For each $\mathsf{a}_\mathsf{k} \in \Sigma$, let $\mathbf{M}_{\mathsf{a}_\mathsf{k}}$ be the $|\mathcal{Q}| \times |\mathcal{Q}|$ matrix gathering the probabilities $M_{\mathsf{a}_\mathsf{k},q_i,q_j} = w_{q_i\to q_j}(\mathsf{a}_\mathsf{k})$ that the transition going from state q_i to state q_j in A outputs the symbol a_k. For notational convenience, we denote this probability by $M_{\mathsf{a}_\mathsf{k}}(q_i, q_j)$.

Now, given a string $\mathsf{s} = \mathsf{s}_1 \ldots \mathsf{s}_\mathsf{t}$, $p(\mathsf{s})$ can be rewritten as follows:

$$p(\mathsf{s}) = p(\mathsf{s}_1 \ldots \mathsf{s}_\mathsf{t}) = \boldsymbol{\tau}^T \mathbf{M}_{\mathsf{s}_1} \cdots \mathbf{M}_{\mathsf{s}_\mathsf{t}} \boldsymbol{\rho} = \boldsymbol{\tau}^T \mathbf{M}_\mathsf{s} \boldsymbol{\rho}, \tag{4.4}$$

where $\boldsymbol{\tau}$ and $\boldsymbol{\rho}$ are two vectors of dimension $|\mathcal{Q}|$ whose components are the values returned by the weight function τ ($\forall q \in S$) and ρ ($\forall q \in F$) respectively, and $\mathbf{M}_{\mathsf{s}} = \mathbf{M}_{\mathsf{s}_1} \cdots \mathbf{M}_{\mathsf{s}_t}$.

From Equation 4.4, we get:

$$\sum_{\mathsf{s} \in \Sigma^*} p(\mathsf{s}) = \sum_{\mathsf{s} \in \Sigma^*} \boldsymbol{\tau}^T \mathbf{M}_{\mathsf{s}} \boldsymbol{\rho}. \tag{4.5}$$

To take into account all possible strings $\mathsf{s} \in \Sigma^*$, Equation 4.5 can be rewritten according to the size of the string s:

$$\sum_{\mathsf{s} \in \Sigma^*} p(\mathsf{s}) = \sum_{i=0}^{\infty} \boldsymbol{\tau}^T (\mathbf{M}_{\mathsf{a}_1} + \mathbf{M}_{\mathsf{a}_2} + \cdots + \mathbf{M}_{\mathsf{a}_{|\boldsymbol{\Sigma}|}})^i \boldsymbol{\rho} = \boldsymbol{\tau}^T \sum_{i=0}^{\infty} \mathbf{M}^{\mathbf{i}} \boldsymbol{\rho}, \tag{4.6}$$

where $\mathbf{M} = \mathbf{M}_{\mathsf{a}_1} + \mathbf{M}_{\mathsf{a}_2} + \cdots + \mathbf{M}_{\mathsf{a}_{|\boldsymbol{\Sigma}|}}$. Denoting $\sum_{i=0}^{\infty} \mathbf{M}^{\mathbf{i}}$ by $\mathbf{B}$, note that

$$\mathbf{B} = \mathbf{I} + \mathbf{M} + \mathbf{M}^{\mathbf{2}} + \mathbf{M}^{\mathbf{3}} + \ldots, \tag{4.7}$$

where $\mathbf{I}$ is the identity matrix. Multiplying $\mathbf{B}$ by $\mathbf{M}$ we get

$$\mathbf{MB} = \mathbf{M} + \mathbf{M}^{\mathbf{2}} + \mathbf{M}^{\mathbf{3}} + \ldots, \tag{4.8}$$

and subtracting Equation 4.7 from Equation 4.8, we get:

$$\mathbf{B} - \mathbf{MB} = \mathbf{I} \Leftrightarrow \mathbf{B} = (\mathbf{I} - \mathbf{M})^{-1}. \tag{4.9}$$

Finally, plugging Equation 4.9 in Equation 4.6, we get a tractable expression for our kernel:

$$K_e(\mathsf{x}, \mathsf{x}') = \sum_{\mathsf{s} \in \Sigma^*} p_e(\mathsf{s}|\mathsf{x}) \cdot p_e(\mathsf{s}|\mathsf{x}') = \boldsymbol{\tau}^T \sum_{i=0}^{\infty} \mathbf{M}^{\mathbf{i}} \boldsymbol{\rho} = \boldsymbol{\tau}^T (\mathbf{I} - \mathbf{M})^{-1} \boldsymbol{\rho}. \tag{4.10}$$

4.3.5 Tractability

In this section, we investigate the complexity of computing $K_e(\mathsf{x}, \mathsf{x}')$ using Equation 4.10 given two strings x and x'. As we have seen in the previous section, this is essentially done by inverting a matrix. Let T be the cPFT modeling the edit probabilities and t its number of states.

The weighted automaton $T|\mathsf{x}$ describing $p_e(\mathsf{s}|\mathsf{x})$ has $t \cdot (|\mathsf{x}| + 1)$ states, and $T|\mathsf{x}'$ describing $p_e(\mathsf{s}|\mathsf{x}')$ has $t \cdot (|\mathsf{x}'| + 1)$ states (see Figure 4.3 for an example). Thus, the matrix $(\mathbf{I} - \mathbf{M})$ has dimension $t^2 \cdot (|\mathsf{x}|+1) \cdot (|\mathsf{x}'|+1)$. The computational cost of each element of this matrix linearly depends on the alphabet size $|\Sigma|$. Therefore, the complexity of computing the

entire matrix is $O(t^4 \cdot |\mathsf{x}|^2 \cdot |\mathsf{x}'|^2 \cdot |\Sigma|)$. Since $\mathbf{M}$ is triangular by construction of $A|\mathsf{x}, \mathsf{x}'$ (the probability of going back to a previous state is zero), the matrix inversion $(\mathbf{I} - \mathbf{M})^{-1}$ can be performed by back substitution, avoiding the complications of general Gaussian elimination. The cost of the inversion is in order of the square of the matrix dimension, that is $O(t^4 \cdot |\mathsf{x}|^2 \cdot |\mathsf{x}'|^2)$. This leads to an overall cost of

$$O(t^4 \cdot |\mathsf{x}|^2 \cdot |\mathsf{x}'|^2 \cdot |\Sigma|).$$

Recall that t stands for the size of the model T. In the case of memoryless models such as that of Oncina & Sebban (2006) used in the experiments, $t = 1$ and thus the complexity is reduced to

$$O(|\mathsf{x}|^2 \cdot |\mathsf{x}'|^2 \cdot |\Sigma|).$$

Therefore, in the case of a memoryless transducer, and for small alphabet sizes, the computational cost of our edit kernel is "only" the square of that of the standard edit distance.

Despite the fact that $\mathbf{M}$ is triangular, the algorithmic complexity remains high when strings are long and/or when the alphabet size is large. In this case, we may approximate our kernel $K_e(\mathsf{x}, \mathsf{x}')$ by computing a finite sum over the training strings $\mathsf{s} \in \mathcal{T}$. Therefore, we get

$$\hat{K}_e(\mathsf{x}, \mathsf{x}') = \sum_{\mathsf{s} \in \mathcal{T}} p_e(\mathsf{s}|\mathsf{x}) \cdot p_e(\mathsf{s}|\mathsf{x}').$$

Since the computational complexity of each probability $p_e(\mathsf{s}|\mathsf{x})$ scales in $|\mathsf{x}| + |\mathsf{s}|$, the average cost of a kernel evaluation is

$$(|\mathsf{x}| + |\mathsf{x}'| + \overline{|\mathsf{s}|}) \cdot |\mathcal{T}|,$$

where $\overline{|\mathsf{s}|}$ is the average length of the training strings.

In conclusion, even if our kernel is rather costly from a complexity point of view, it can be derived from any transducer modeling edit probabilities, and may be approximated if needed. In the next section, we provide experimental evidence that our kernel outperforms standard and learned edit distances plugged in k-NN as well as standard string kernels.

4.4 Experimental Validation

4.4.1 Setup

To assess the relevance of our string edit kernel, we carry out experiments on the well-known NIST Special Database 3 of the National Institute of Standards and Technology,

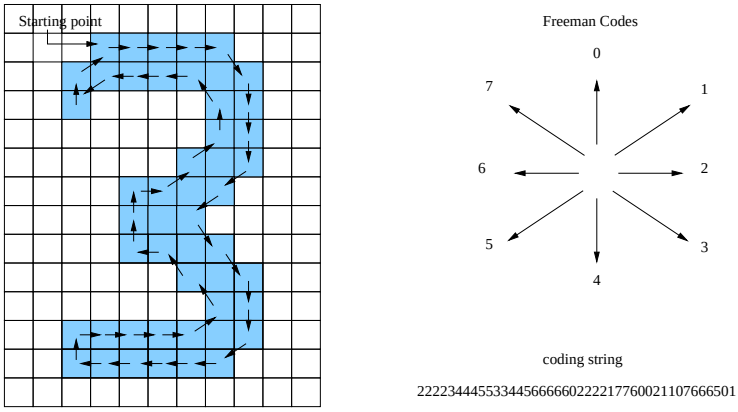


FIGURE 4.5: A handwritten digit and its string representation.

which is a handwritten character dataset.

We focus on the set of 10,000 handwritten digits given as 128×128 bitmap images. We use a training sample of about $8,000$ instances and a test sample of $2,000$ instances. Each instance is represented by a string of Freeman codes (Freeman, 1974). To encode a digit, the algorithm scans the bitmap from left to right, starting from the top until reaching the first pixel of the digit. It then follows the contour of the digit until it returns to the starting pixel. The string coding the digit is the sequence of Freeman codes representing the successive directions of the contour. Figure 4.5 shows an example of this encoding procedure.

We use SVM-Light[2] as the SVM implementation to compare our approach with other string kernels and adopt a one-versus-all approach to deal with the multi-class setting. This consists in learning a model M_i for each class, where M_i is learned from a positive class made of digits labeled i and a negative class made of differently labeled digits. Then, the class of a test instance x is determined as follows: we compute the margin $M_i(\mathsf{x})$ for each model M_i. A high positive value of $M_i(\mathsf{x})$ represents a high probability for x to be of class i. The predicted class of x is given by $\arg\max_i M_i(\mathsf{x})$.

4.4.2 Comparison with Edit Distances

As done by Neuhaus & Bunke (2006), our first objective is to compare our edit kernel K_e with edit distances used in a k-NN algorithm. We use two edit distances: (i) the standard Levenshtein edit distance d_{lev} with all costs set to 1, and (ii) a stochastic edit dissimilarity $d_e(\mathsf{x}, \mathsf{x}') = -\log p_e(\mathsf{x}'|\mathsf{x})$ learned with SEDiL (Boyer et al., 2008), a software that implements (among others) the method of Oncina & Sebban (2006).

We assess the performance of a 1-nearest neighbor algorithm using d_{lev} and d_e, and compare them with our string edit kernel plugged in a SVM classifier. Note that the

[2] http://svmlight.joachims.org/

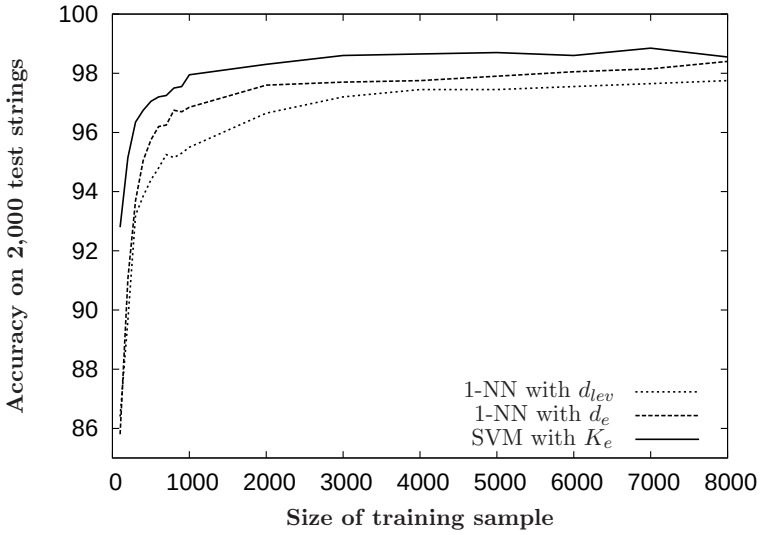


FIGURE 4.6: Comparison of our edit kernel with edit distances on a handwritten digit recognition task.

Training sample size	**1,000**	**2,000**	**3,000**	**4,000**	**5,000**	**6,000**	**7,000**	**8,000**
K_e vs d_{lev}	**6E-06**	**4E-04**	**1E-03**	**3E-03**	**2E-03**	**8E-03**	**2E-03**	**3E-02**
K_e vs d_e	**1E-02**	6E-02	**2E-02**	**2E-02**	**2E-02**	9E-02	**3E-02**	3E-01

TABLE 4.1: Statistical comparison of our edit kernel with standard and learned edit distances (p-values of a Student's paired t-test). Boldface indicates that the difference is significant in favor of our kernel using a risk of 5%.

conditional edit probabilities $p_e(\mathsf{x}'|\mathsf{x})$ used in our edit kernel are the same as those used in $d_e(\mathsf{x}, \mathsf{x}')$. Results are shown in Figure 4.6 with respect to an increasing number of training instances (from 100 to $8,000$).

We can make the following remarks:

- First, learning an edit distance d_e on this classification task leads to better results than using the standard edit distance d_{lev}. Indeed, the accuracy of d_e is always higher than that of d_{lev} regardless of the size of the training sample.
- Second, K_e outperforms both the standard edit distance d_{lev} and the learned edit distance d_e for all training sample sizes. This highlights the usefulness of our kernel.

We estimate the statistical significance of these results using a Student's paired t-test. Table 4.1 contains the p-values obtained when comparing our kernel with d_{lev} and d_e. Using a risk of 5%, the difference is almost always significant in favor of our kernel (shown in boldface in the table).

These results are positive but not quite fair since our edit kernel is plugged into a SVM classifier while the edit distances are plugged into a k-NN classifier. In the next section, we compare K_e with other string kernels of the literature.

4.4.3 Comparison with Other String Kernels

In this second series of experiments, we compare K_e with:

- two classic string kernels, the spectrum kernel (Leslie et al., 2002a) and the subsequence kernel (Lodhi et al., 2002),
- a variant of the edit kernel of Li & Jiang (2004) based on learned edit probabilities:[3]

$$K_{L\&J}(\mathsf{x}', \mathsf{x}) = e^{\frac{1}{2}t(\log p_e(\mathsf{x}'|\mathsf{x}) + \log p_e(\mathsf{x}|\mathsf{x}'))},$$

- and the edit kernel $K_{N\&B}$ (Neuhaus & Bunke, 2006) in its original version since it cannot accommodate p_e in a straightforward way.

Recall that these kernels were presented in Section 2.3.3. We did not include the local alignment kernel (Saigo et al., 2004) in this experimental study since it is based on local alignments and is specific to finding remote homologies in protein sequences.

The parameter p specifying the length of the considered subsequences in the spectrum and the subsequence kernel was set to 2. The subsequence kernel also has a parameter λ which is used to give less importance to subsequences with large gaps. We set λ to 2. The parameter t of $K_{L\&J}$ was set to 0.02. These parameter values give the best results on the dataset.

Figure 4.7 shows the results we obtain with the considered kernels. We first note that the best results are obtained with edit kernels. As in the previous experiment, Table 4.2 gives the p-values of the Student's t-test. Our edit kernel significantly outperforms all other string kernels except $K_{L\&J}$: both kernels perform comparably, and the difference for a given training sample size is not significant. However, K_e gives slightly better results for most training sample sizes (12 times out of 17): if a sign test is used, this yields a p-value of 0.07, indicating that the difference is significant with a risk of 7%.

It is also important to keep in mind that $K_{L\&J}$ is not guaranteed to be a valid kernel and thus the parameter t must be tuned with care. Figure 4.8 demonstrates that this kernel can perform poorly if t is not tuned properly. Unlike $K_{L\&J}$, our edit kernel is guaranteed to be valid and is parameter-free.

[3] $K_{L\&J}$ is made symmetric by adding $p_e(\mathsf{x}'|\mathsf{x})$ and $p_e(\mathsf{x}|\mathsf{x}')$.

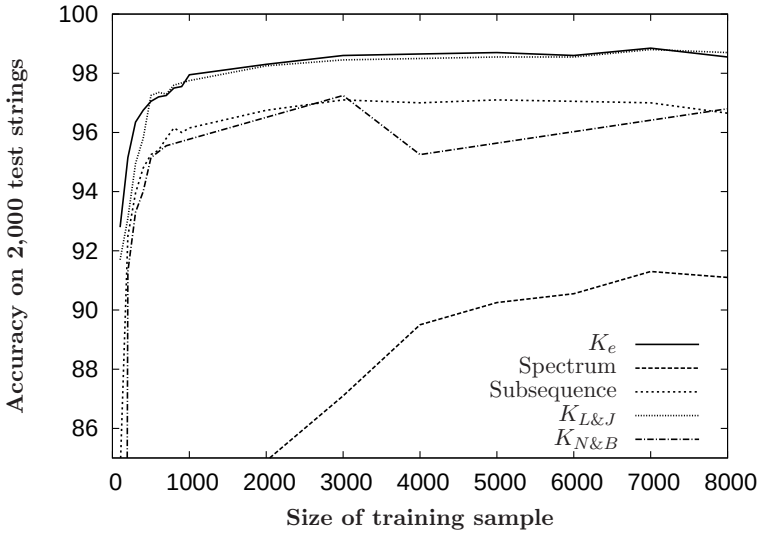


FIGURE 4.7: Comparison of our edit kernel with other string kernels on a handwritten digit recognition task.

Training sample size	**1,000**	**2,000**	**3,000**	**4,000**	**5,000**	**6,000**	**7,000**	**8,000**
K_e vs spectrum	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
K_e vs subsequence	**4E-04**	**8E-04**	**5E-04**	**2E-04**	**2E-04**	**4E-04**	**2E-05**	**4E-05**
K_e vs $K_{L\&J}$	3E-01	4E-01	3E-01	3E-01	3E-01	4E-01	4E-01	7E-01
K_e vs $K_{N\&B}$	**6E-06**	**6E-06**	**1E-03**	**2E-10**	**6E-09**	**4E-08**	**4E-07**	**1E-04**

TABLE 4.2: Statistical comparison of our edit kernel with other string kernels (p-values of a Student's paired t-test). Boldface indicates that the difference is significant in favor of our kernel using a risk of 5%.

4.5 Conclusion

In this chapter, we designed a new string edit kernel that can make use of edit probabilities learned with generative or discriminative probabilistic models while enjoying the classification performance brought by SVM. We showed that although it involves an infinite sum over an entire language, our kernel can be computed exactly through the intersection of probabilistic automata built from the edit probability model and a matrix inversion. Experiments on a handwritten digit recognition task have shown that our edit kernel outperforms standard and learned edit distance within a k-NN framework as well as state-of-the-art string kernels.

An interesting perspective is to improve the algorithmic complexity of our kernel. The main bottleneck in its calculation is the size of the intersection automaton that allows the computation of $p_e(\mathsf{s}|\mathsf{x})p_e(\mathsf{s}|\mathsf{x}')$. A way of reducing its size could consist in simplifying the conditional transducers $T|\mathsf{x}$ and $T|\mathsf{x}'$ from which it is built by only considering the most likely transitions and states. A simplification of these automata would have a

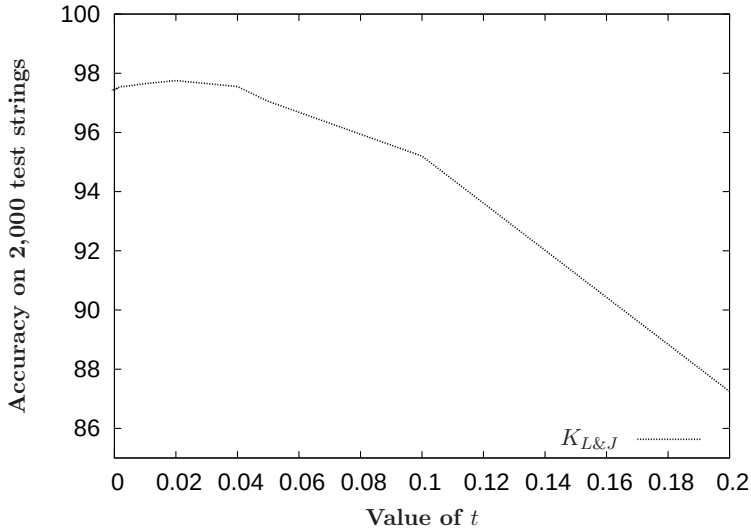


FIGURE 4.8: Influence of the parameter t of $K_{L\&J}$ (1,000 training strings).

direct impact on the dimension of the matrix that has to be inverted, and thus on the evaluation cost of the kernel.

A second perspective is to extend this work to the design of tree edit kernels. Indeed, as seen in Section 3.3.2, generative and discriminative models for learning tree edit probabilities have been proposed and could be used to derive powerful tree edit kernels, based on the same ideas as in the string case.

While one of the advantages of the presented approach is to incorporate a lot of structural information by comparing inputs strings to an infinite number of strings, it makes it difficult to establish generalization guarantees. In the next chapter, we overcome this limitation by proposing a novel edit similarity learning approach that is not subject to many classic limitations of previous edit metric learning methods (in particular, those based on probabilistic models that our kernel uses) and for which we can derive a generalization bound. The idea is to relax the structural constraint on edit scripts to get an edit similarity that has a simpler form and can thus be learned through numerical optimization. The resulting (potentially non-PSD) similarity can then be used directly to build a linear classifier (that has bounded true risk), avoiding the computational cost of transforming it into a kernel. Furthermore, the linear classifiers are sparser than SVM models, speeding up prediction.

CHAPTER 5

Learning Good Edit Similarities from Local Constraints

Chapter abstract Metrics based on the edit distance are widely used to tackle problems involving string or tree-structured data. Unfortunately, as seen in Chapter 4, using them in kernel methods is often difficult and/or costly. On the other hand, the recently-proposed theory of (ϵ, γ, τ)-good similarity functions bridges the gap between the properties of a non-PSD similarity function and its performance in linear classification. In this chapter, we show that this framework is well-suited to edit similarities. Furthermore, we make use of a relaxation of (ϵ, γ, τ)-goodness to propose a novel edit similarity learning method, GESL, that avoids the classic drawbacks of previous approaches. Using uniform stability, we derive generalization bounds that hold for a large class of loss functions and show that they can be related to the error of a linear classifier built from the similarity. We also provide experimental results on two real-world datasets highlighting that edit similarities learned with GESL induce more accurate and sparser classifiers than other (standard or learned) edit similarities.

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

Aurélien Bellet, Amaury Habrard, and Marc Sebban. An Experimental Study on Learning with Good Edit Similarity Functions. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 126–133, 2011a.

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, 2011c.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. Good edit similarity learning by loss minimization. *Machine Learning Journal (MLJ)*, 89(1):5–35, 2012b.

5.1 Introduction

As mentioned in the previous chapter, metrics based on the edit distance are widely used by practitioners when dealing with string or tree-structured data. Although they

involve complex procedures, there exist a few methods (reviewed in Section 3.3) for learning edit metrics for a given task. These edit metrics are typically used in a k-NN setting. As we have seen in Chapter 4, using them in kernel methods such as SVM requires the design of a positive semi-definite edit kernel. However, existing edit kernels are either not guaranteed to be PSD, or involve rather costly procedures (Li & Jiang, 2004; Neuhaus & Bunke, 2006; Bellet et al., 2010). Furthermore, there is a lack of theoretical understanding of how arbitrary similarity functions can be used to learn accurate linear classifiers.

Recently, Balcan et al. (2006; 2008a; 2008b) introduced a theory of learning with so-called (ϵ, γ, τ)-good similarity functions that gives intuitive, sufficient conditions for a similarity function to allow one to learn well. Essentially, a similarity function K is (ϵ, γ, τ)-good if a $1-\epsilon$ proportion of examples are on average more similar to *reasonable* examples of the same class than to *reasonable* examples of the opposite class by a margin γ, where a τ proportion of examples must be *reasonable*. K does not have to be a metric nor positive semi-definite (PSD). They show that if K is (ϵ, γ, τ)-good, then it can be used to build a linear separator in an explicit projection space that has margin γ and error arbitrarily close to ϵ. This separator can be learned efficiently using a linear program and tends to be sparse thanks to L_1 norm regularization.

The first contribution of this work is to experimentally show that this theory is well-suited to edit similarity functions and is competitive with SVM in terms of accuracy, while inducing sparser models. Furthermore, we show that we can make use of this framework to propose a new approach to learning string and tree edit similarities which addresses the classic drawbacks of other methods in the literature, i.e., lack of generalization guarantees, high computational cost, convergence to suboptimal solution and inability to use the information brought by negative pairs. Our approach (GESL, for Good Edit Similarity Learning) is driven by the idea of (ϵ, γ, τ)-goodness: we learn the edit costs so as to optimize a relaxation of the goodness of the resulting similarity function. It is based on regularized risk minimization (formulated as an efficient convex program) over some positive and negative training pairs: the similarity is thus optimized with respect to *local* constraints but plugged in a *global* linear classifier. We provide an extensive theoretical study of the properties of GESL based on a notion of uniform stability adapted to metric learning (Jin et al., 2009), leading to the derivation of a generalization bound that holds for a large class of loss functions. This bound can be related to the generalization error of the linear classifier built from the similarity and is independent of the size of the alphabet, making GESL suitable for handling problems with large alphabet. To the best of our knowledge, this is the first edit metric learning method with generalization guarantees, and the first attempt to establish a theoretical relationship between a learned metric and the risk of a classifier using it. We show in a comparative experimental study that GESL has fast convergence and leads to more accurate and sparser classifiers than other (standard or learned) edit similarities.

The rest of this chapter is organized as follows. In Section 5.2, we introduce the theory of (ϵ, γ, τ)-goodness. Section 5.3 features a preliminary study that provides experimental evidence that this theory is well-suited to edit similarity functions and leads to classifiers that are competitive with SVM classifiers. Section 5.4 presents GESL, our approach to learning (ϵ, γ, τ)-good edit similarities. We show that it is a suitable way to deal not only with strings but also with tree-structured data. We propose in Section 5.5 a theoretical analysis of GESL based on uniform stability, leading to the derivation of a generalization bound. We also provide a discussion on that bound and its implications, as well as a way of deriving a bound for the case where instances have unbounded size. A wide experimental evaluation of our approach on two real-world string datasets from the natural language processing and image classification domains is provided in Section 5.6. Finally, we conclude this work in Section 5.7.

5.2 The Theory of (ϵ, γ, τ)-Good Similarity Functions

In recent work, Balcan et al. (2006; 2008a; 2008b) introduced a new theory of learning with good similarity functions. Their motivation was to overcome two major limitations of kernel theory. First, a good kernel is essentially a good similarity function, but the theory talks in terms of margin in an implicit, possibly unknown projection space, which can be a problem for intuition and design. Second, the PSD and symmetry requirement often rules out natural similarity functions for the problem at hand. As a consequence, Balcan et al. (2008b) proposed the following definition of good similarity function.

Definition 5.1 (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 $R(x)$ defining a (probabilistic) set of "reasonable points" such that the following conditions hold:

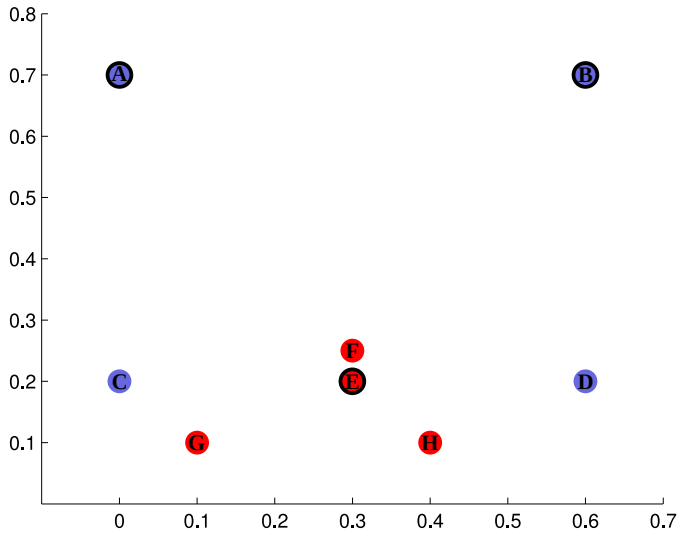
1. A $1 - \epsilon$ probability mass of examples (x, y) satisfy

$$\mathbb{E}_{(x',y')\sim P}[yy'K(x, x')|R(x')] \geq \gamma, \tag{5.1}$$

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

The first condition is essentially requiring that *a $1 - \epsilon$ proportion of examples x are on average more similar to reasonable examples of the same class than to reasonable examples of the opposite class by a margin γ* and the second condition that *at least a τ proportion of the examples are reasonable.*[1] Figure 5.1 illustrates the definition on a toy example. Note that other definitions are possible, like those proposed by Wang et al.

[1]For now, we assume that the set of reasonable points is given. The question of finding such a set is addressed later in this section.



	A	B	C	D	E	F	G	H
A	1	0.40	0.50	0.22	0.42	0.46	0.39	0.28
B	0.40	1	0.22	0.50	0.42	0.46	0.22	0.37
E	0.42	0.42	0.70	0.70	1	0.95	0.78	0.86
Margin	0.3277	0.3277	0.0063	0.0063	0.0554	0.0106	0.0552	0.0707

FIGURE 5.1: A graphical insight into Definition 5.1. Let us consider 8 points as shown above (blue represents the positive class, red the negative class) and use the similarity function $K(\mathbf{x}, \mathbf{x}') = 1 - \|\mathbf{x} - \mathbf{x}'\|_2$. We picked 3 reasonable points (A, B and E, circled in black), thus we can set $\tau = 3/8$. Similarity scores to the reasonable points as well as the margin achieved by each point (as given by Equation 5.1) are shown in the array. There exists an infinite number of valid instantiations of ϵ and γ since there is a trade-off between the margin γ and proportion of margin violations ϵ. For example, K is $(0, 0.006, 3/8)$-good because all points ($\epsilon = 0$) are on average more similar to reasonable examples of the same class than to reasonable examples of the other class by a margin $\gamma = 0.006$. One can also say that K is $(2/8, 0.01, 3/8)$-good ($\epsilon = 2/8$ because examples C and D violate the margin $\gamma = 0.01$).

(2007, 2009) for unbounded dissimilarity functions. Yet Definition 5.1 is very interesting in three respects. First, it is a strict generalization of the notion of good kernel (Balcan et al., 2008b) but does not impose positive semi-definiteness nor symmetry. Second, as opposed to pair and triplet-based criteria used in metric learning, Definition 5.1 is based on an average over some points. In other words, it relaxes the notion of local constraints, opening the door to metric learning for global algorithms. Third, these conditions are sufficient to learn well, i.e., to induce a classifier with low true risk, as we show in the following.

Let K be an (ϵ, γ, τ)-good similarity function. If the set of reasonable points $R = \{(x'_1, y'_1), (x'_2, y'_2), \ldots, (x'_{|R|}, y'_{|R|})\}$ is known, it follows directly from Equation 5.1 that

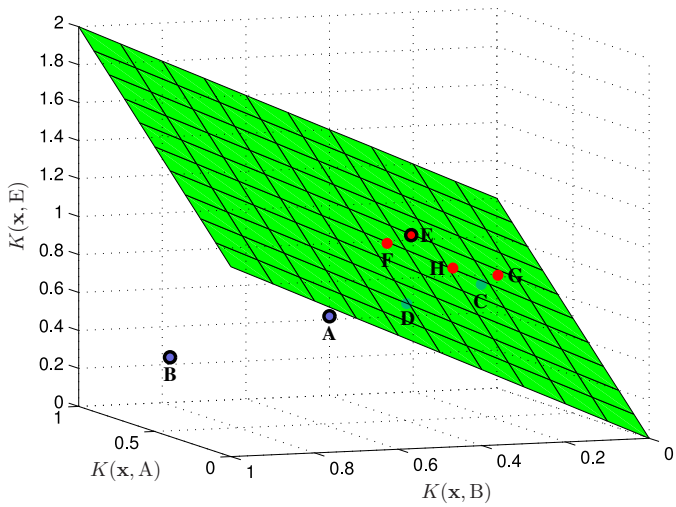


FIGURE 5.2: Projection space (ϕ-space) implied by the toy example of Figure 5.1: similarity scores to the reasonable points (A, B and E) are used as new features. Since K is $(0, \gamma, 3/8)$ for some $\gamma > 0$, the linear separator of equation $K(\mathbf{x}, \mathrm{A}) + K(\mathbf{x}, \mathrm{B}) - K(\mathbf{x}, \mathrm{E}) = 0$ (shown as a green grid) achieves perfect classification, although the data were not linearly separable in the original space.

the following classifier achieves true risk at most ϵ at margin γ:

$$h(x) = \text{sign}\left[\frac{1}{|R|}\sum_{i=1}^{|R|} y'_i K(x, x'_i)\right].$$

Note that h is a linear classifier in the space of the similarity scores to the reasonable points. In other words, K is used to project the data into a new space using the mapping $\phi : \mathcal{X} \to \mathbb{R}^{|R|}$ defined as:

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

The projection and linear classifier corresponding to the toy example of Figure 5.1 is shown in Figure 5.2.

However, in practice the set of reasonable points is unknown. We can get around this problem by sampling points (called *landmarks*) and use them to project the data into a new space (using the same strategy as before).[2] If we sample enough landmarks (this depends in particular on τ, which defines how likely it is to draw a reasonable point), then with high probability there exists a linear classifier in that space that achieves true risk close to ϵ. This is formalized in Theorem 5.2.

Theorem 5.2 (Balcan et al., 2008b)**.** *Let K be an (ϵ, γ, τ)-good similarity function for a learning problem P. Let $\mathcal{L} = \{x'_1, x'_2, \ldots, x'_{n_\mathcal{L}}\}$ be a sample of $n_\mathcal{L} = \frac{2}{\tau}\left(\log(2/\delta) + 8\frac{\log(2/\delta)}{\gamma^2}\right)$ landmarks drawn from P. Consider the mapping $\phi^\mathcal{L} : \mathcal{X} \to \mathbb{R}^{n_\mathcal{L}}$ defined as follows:*

[2]Note that the landmark points need not to be labeled, although we do not make use of this feature in our contributions.

$\phi_i^{\mathcal{L}}(x) = K(x, x'_i)$, $i \in \{1, \ldots, n_{\mathcal{L}}\}$. *Then, with probability at least* $1-\delta$ *over the random sample* $\mathcal{L}$*, the induced distribution* $\phi^{\mathcal{L}}(P)$ *in* $\mathbb{R}^{n_{\mathcal{L}}}$ *has a linear separator of error at most* $\epsilon + \delta$ *relative to* L_1 *margin at least* $\gamma/2$.

Unfortunately, finding this separator is NP-hard (even to approximate) because minimizing the number of L_1 margin violations is NP-hard. To overcome this limitation, the authors considered the hinge loss as a surrogate for the 0/1 loss (which counts the number of margin violations) in the following reformulation of Definition 5.1.

Definition 5.3 (Balcan et al., 2008b)**.** A similarity function K is an (ϵ, γ, τ)-*good similarity function in hinge loss* for a learning problem P if there exists a (random) indicator function $R(x)$ defining a (probabilistic) set of "reasonable points" such that the following conditions hold:

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

This leads to the following theorem, similar to Theorem 5.2.

Theorem 5.4 (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* $0 \leq \delta \leq \gamma\epsilon_1/4$*, let* $\mathcal{L} = \{x'_1, x'_2, \ldots, x'_{n_{\mathcal{L}}}\}$ *be a sample of* $n_{\mathcal{L}} = \frac{2}{\tau}\left(\log(2/\delta) + 16\frac{\log(2/\delta)}{\epsilon_1\gamma^2}\right)$ *landmarks drawn from* P*. Consider the mapping* $\phi^{\mathcal{L}} : X \to \mathbb{R}^{n_{\mathcal{L}}}$ *defined as follows:* $\phi_i^{\mathcal{L}}(x) = K(x, x'_i)$, $i \in \{1, \ldots, n_{\mathcal{L}}\}$*. Then, with probability at least* $1-\delta$ *over the random sample* $\mathcal{L}$*, the induced distribution* $\phi^{\mathcal{L}}(P)$ *in* $\mathbb{R}^{n_{\mathcal{L}}}$ *has a linear separator of error at most* $\epsilon + \epsilon_1$ *at margin* γ.

The objective is now to find a linear separator $\boldsymbol{\alpha} \in \mathbb{R}^{n_{\mathcal{L}}}$ that has low true risk based on the expected hinge loss relative to L_1 margin γ:

$$\mathbb{E}_{(x,y)\sim P}\left[\left[1 - y\left\langle \boldsymbol{\alpha}, \phi^{\mathcal{L}}(x)\right\rangle/\gamma\right]_+\right].$$

Using a landmark sample $\mathcal{L} = \{x'_1, x'_2, \ldots, x'_{n_{\mathcal{L}}}\}$ and a training sample $\mathcal{T} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, one can find this separator $\boldsymbol{\alpha}$ efficiently by solving the following linear program (LP):[3]

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \left[1 - \sum_{j=1}^{n_{\mathcal{L}}} \alpha_j y_i K(x_i, x'_j)\right]_+ + \lambda\|\boldsymbol{\alpha}\|_1. \tag{5.2}$$

In practice, we simply use the training examples as landmarks. In this case, learning rule (5.2) — referred to as "Balcan's learning rule" in the rest of this document — is

[3]The original formulation (Balcan et al., 2008b) was actually L_1-constrained. We provide here an equivalent, more practical L_1-regularized form.

reminiscent of the standard SVM formulation, with three important differences. First, recall that K is not required to be PSD nor symmetric. Second, the linear classifier lies in an explicit projection space built from K (called an empirical similarity map) rather than in a possibly implicit Hilbert Space induced by a kernel. Third, it uses L_1 regularization, inducing sparsity in $\boldsymbol{\alpha}$ and thus reducing the number of landmarks the classifier is based on, which speeds up prediction.[4] This regularization can be interpreted as a way to select (or approximate) the set of reasonable points among the landmarks: in a sense, R is automatically worked out while learning $\boldsymbol{\alpha}$.[5] Note that we can control the degree of sparsity of the linear classifier: the larger λ, the sparser $\boldsymbol{\alpha}$.

To sum up, the performance of the linear classifier theoretically depends on how well the similarity function satisfies Definition 5.1. In this chapter, we first conduct a preliminary experimental study to investigate the level of (ϵ, γ, τ)-goodness of some edit similarities and their performance in classification when used in Balcan's learning rule (Section 5.3). The rest of the chapter is the main contribution and is devoted to learning (ϵ, γ, τ)-good edit similarities from data.

5.3 Preliminary Experimental Study

In this section, we experimentally show that the framework of (ϵ, γ, τ)-goodness is well-suited to edit similarities. We first investigate the goodness of edit similarities on the previously-studied handwritten digit recognition task (Section 5.3.1). Then, in Section 5.3.2, we compare the performance of linear classifiers learned with Balcan's rule using edit similarities with the performance of SVM using standard edit kernels.

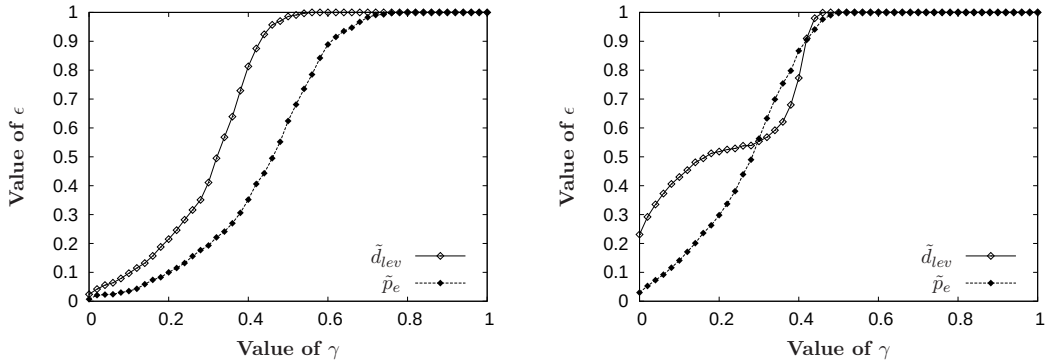
5.3.1 Are Edit Similarities Really (ϵ, γ, τ)-Good?

In this experimental evaluation of the (ϵ, γ, τ)-goodness of edit similarities, we will consider the standard Levenshtein distance d_{lev} and edit probabilities p_e learned with the method of Oncina & Sebban (2006). We actually use $-d_{lev}$ so that both similarities express a measure of closeness, making the comparison easier. We also normalized them so that they lie in $[-1, 1]$.[6] In the following, they are referred to as $\tilde{d}_{lev}$ and $\tilde{p}_e$. Looking at Definition 5.1, we can easily estimate ϵ, γ and τ using a randomly selected set of points. We illustrate this on the NIST Special Database 3, the handwritten digit recognition

[4]Note that L_1 regularization has also been used in the context of standard SVM formulations, leading to the 1-norm SVM (Zhu et al., 2003). While these classifiers may work well in practice, most of the handy SVM theory fall apart in this case. Conversely, the use of (5.2) is justified by the theory presented in this section.

[5]The problem of finding the reasonable points is not as simple if we first want to *learn* the similarity function, as we will see later in this chapter.

[6]We normalized them to zero mean and unit variance, then brought back to 1 and -1 the values greater than 1 and smaller than -1 respectively. We are aware that there may be better normalizations but this is outside the scope of this work.



(a) 0 vs. 1 (b) 0 vs. 8

FIGURE 5.3: Estimation of ϵ as a function of γ for $\tilde{d}_{lev}$ and $\tilde{p}_e$ on two handwritten digits binary classification tasks.

task already used in Chapter 4, where the digits are represented as strings of Freeman codes.

Since we do not know the set of reasonable points before learning the linear classifier, we fix $\tau = 1$ (i.e., all points are considered reasonable) and plot ϵ as a function of γ. In order to analyze the results in different contexts, we randomly selected 500 instances of each class and estimated the goodness of the similarities for each binary problem. For brevity, we only discuss the goodness curves for two representative problems: "0 vs. 1" and "0 vs. 8", shown in Figure 5.3. The interpretation (given by Definition 5.1) is that a margin γ leads to an ϵ proportion of examples violating the margin. For the "0 vs. 1" problem, shown in Figure 5.3(a), both similarities achieve good margin while keeping the number of violations small. The learned similarity $\tilde{p}_e$ behaves slightly better. The "0 vs. 8" problem is a harder task, since the representation of an eight is often similar to that of a zero (because Freeman codes only encode the contour of the digits). Figure 5.3(b) reflects the difficulty of the task, since margin violations are almost always higher for a given γ than in the "0 vs. 1" case. For the "0 vs. 8" task, the learned similarity provides an important improvement over the standard edit distance: for small margin values, it achieves few margin violations.

To sum up, we see that decent values for γ and ϵ are achieved even without selecting an appropriate subset $\mathcal{R}$ of reasonable points. Note that we observe a similar behavior for all binary problems in the dataset. Therefore, edit similarities satisfy Definition 5.1 rather well, thus Theorem 5.4 is meaningful and we can expect good accuracy in linear classification on this dataset. Moreover, $\tilde{p}_e$ seems to be "(ϵ, γ, τ)-better" than $\tilde{d}_{lev}$, which suggests that it could achieve better generalization performance. We will see that it is indeed the case in the next section.

5.3.2 Experiments

In this section, we provide experimental evidence that learning with Balcan's learning rule using edit similarities outperforms a k-NN approach and is competitive with a standard SVM approach, while inducing much sparser models. As noted earlier, standard SVM and Balcan's learning rule are similar but use different regularizers (L_2 norm and L_1 norm respectively). The comparative performance of L_2 and L_1 regularized learning rules has been the subject of previous experimental studies (see for instance Zhu et al., 2003) but, to the best of our knowledge, never in the context of edit similarities. Furthermore, (ϵ, γ, τ)-goodness provides a theoretical justification of Balcan's learning rule and casts an interesting light on this comparison.[7]

We compare the following approaches: (i) Balcan's learning rule (5.2) using $K(\mathsf{x}, \mathsf{x}') = \tilde{d}_{lev}(\mathsf{x}, \mathsf{x}')$, (ii) Balcan's learning rule using $K(\mathsf{x}, \mathsf{x}') = \tilde{p}_e(\mathsf{x}'|\mathsf{x})$, (iii) SVM learning using $K(\mathsf{x}, \mathsf{x}') = e^{-t \cdot d_{lev}(\mathsf{x}, \mathsf{x}')}$, the kernel of Li & Jiang (2004) based on d_{lev}, (iv) SVM learning using $K(\mathsf{x}, \mathsf{x}') = e^{\frac{1}{2} t (\log p_e(\mathsf{x}'|\mathsf{x}) + \log p_e(\mathsf{x}|\mathsf{x}'))}$, the kernel of Li & Jiang (2004) based on p_e, (v) 1-NN using $d_{lev}(\mathsf{x}, \mathsf{x}')$, and (vi) 1-NN using $-p_e(\mathsf{x}'|\mathsf{x})$. We choose LIBSVM[8] as the SVM implementation, which takes a one-versus-one approach for multi-class classification. We thus use the same strategy for multi-class classification with Balcan's learning rule. Note that we take the training examples to be the landmarks. Therefore, all learning algorithms have access to strictly the same information (that is, similarity measurements between training examples), allowing a fair comparison.

In the following, we present results on the multi-class handwritten digit classification task and on a dataset of English and French words.

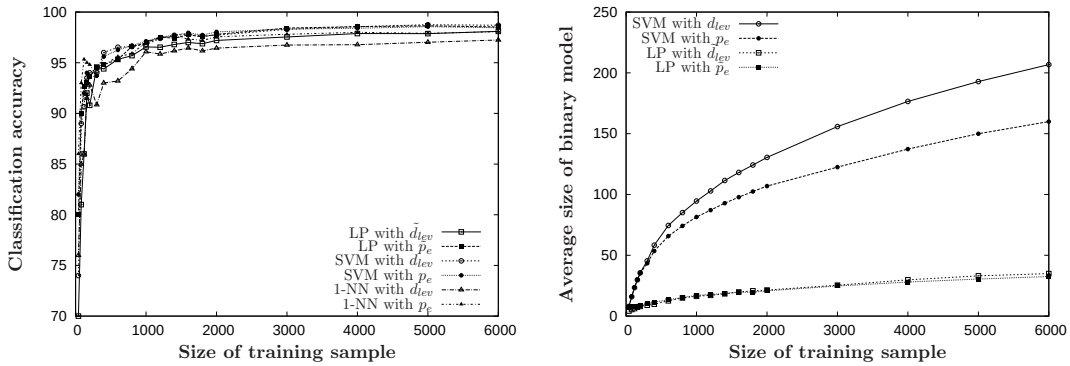
5.3.2.1 Handwritten digit classification

Using the handwritten digit classification dataset, we first aim at evaluating the performance of the models obtained with different methods. We use 40 to 6,000 training examples, reporting the results under 5-fold cross-validation. The parameters of the models, such as λ for approaches (i-ii) or C and t for approaches (iii-iv), are tuned by cross-validation on an independent set of examples, always selecting the value that offers the best classification accuracy.

Accuracy and sparsity Classification accuracy is reported in Figure 5.4(a). All methods perform essentially the same, except for 1-NN that is somewhat weaker. Note that the methods based on the learned edit probabilities are, as expected, more accurate than those based on the standard edit distance. Figure 5.4(b) shows the average size of

[7] Note that we did not include 1-norm SVM in this experimental study because the learning rule itself is very similar to Balcan's learning rule while having no grounds in SVM theory.

[8] http://www.csie.ntu.edu.tw/~cjlin/libsvm/



(a) 0 vs. 1 (b) 0 vs. 8

FIGURE 5.4: Classification accuracy and sparsity results for methods (i-vi) over a range of training set sizes (Digit dataset).

a binary model for approaches (i-iv), i.e., the number of training examples (reasonable points or support vectors) involved in the classification of new examples. Approaches (i-ii) are 5 to 6 times sparser than (iii-iv), which confirms that learning with Balcan leads to much sparser models than standard SVM learning.

Influence of the parameters We now study the influence of parameters on the accuracy and sparsity of the models. Results are obtained on 4,000 training examples. The influence of λ on the models learned with Balcan is shown in Figure 5.5. The results confirm that λ can be conveniently used to control the sparsity of the models thanks to L_1 regularization. It is worth noting that while the best accuracy is obtained with relatively small values ($\lambda \in [1; 10]$), one can get even sparser but still very accurate models with larger values ($\lambda \in [10; 200]$). This is especially true when using $\tilde{p}_e$. Therefore, one can learn a model with Balcan that is just slightly less accurate than the corresponding SVM model while being 10 to 18 times sparser. This can be a useful feature, in particular in applications where data storage is limited and/or high classification speed is required.

We also investigate the influence of parameter t on the performance of the SVM models. Results are shown in Figure 5.6 (a log-scale is used to allow a better appreciation of the variations). Both the accuracy and the sparsity of the SVM models are heavily dependent on t: only a narrow range of t values (probably those achieving positive semi-definiteness) allows for accurate and acceptably-sized models. Furthermore, this range appears to be specific to the edit similarity used. Therefore, t must be tuned very carefully, which represents a waste of time and data.

Lastly, one might wonder whether the SVM parameter C can also be used to improve the sparsity of the models in the same way as λ. In order to assess this, we try a wide range of C values and record the average sparsity of the models. SVM could not match the sparsity of the models learned with Balcan. The best average size for a binary model

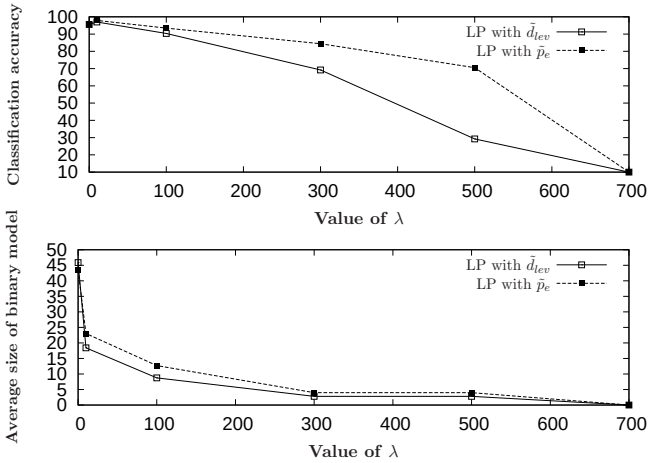


FIGURE 5.5: Classification accuracy and sparsity results with respect to the value of λ (Digit dataset).

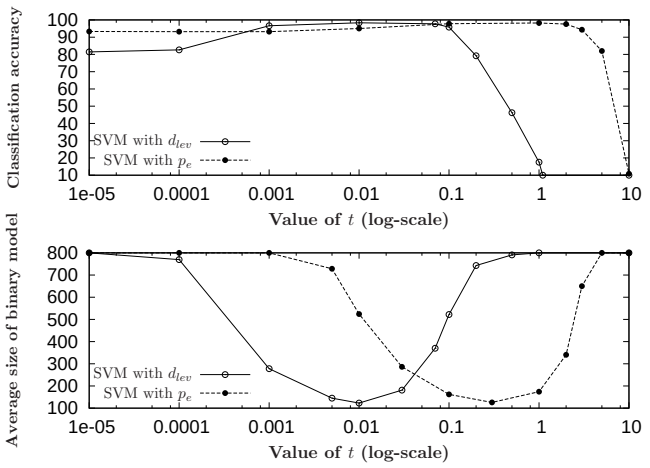


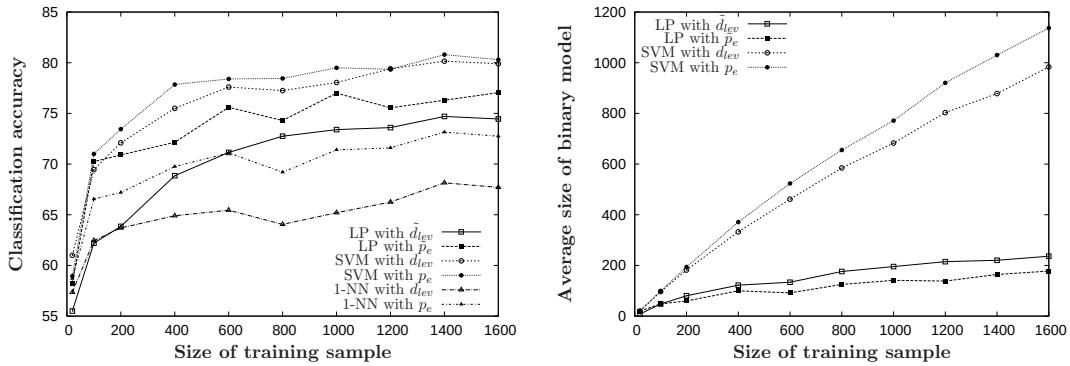
FIGURE 5.6: Classification accuracy and sparsity results with respect to the value of t in log-scale (Digit dataset).

was greater than 100, i.e., more than 2 times bigger than the worst model size obtained with Balcan. This results from the tendency of L_2 regularization to select models that put small weights on many coordinates.

5.3.2.2 English and French words classification

In this second series of experiments, we choose a different and harder task: classifying words as either English or French. We use the 2,000 top words lists from Wiktionary.[9] We only consider unique words (i.e., not appearing in both lists) of length at least 4, and we also get rid of accent and punctuation marks. We end up with about 2,600

[9] http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists



(a) Accuracy (b) Sparsity

FIGURE 5.7: Word dataset: classification accuracy and sparsity results for methods (i-vi) over a range of training set sizes.

words. We keep 600 words aside for cross-validation of parameters, 400 words to test the models and use the remaining words to learn the models.

Classification accuracy is reported in Figure 5.7(a). Note that this binary task is significantly harder than the one presented in the previous section, and that once again, models based on p_e perform better than those based on d_{lev}. Models learned with Balcan clearly outperform k-NN, while SVM models are the most accurate. Sparsity results are shown in Figure 5.7(b). The gap in sparsity between models learned with Balcan and SVM models is even greater on this dataset: the number of support vectors grows almost linearly with the number of training examples. This is consistent with the theoretical rate established by Steinwart (2003).

5.3.3 Conclusion

In this section, we have shown that edit similarities fit the framework of (ϵ, γ, τ)-goodness and that the performance is competitive with standard SVM, with the additional advantages that arbitrary (in particular, non-PSD) similarities can be used and that the classifiers are sparser. We have also seen that this series of experiments confirms the theoretical dependence between the (ϵ, γ, τ)-goodness of the edit similarity function and its performance in classification.

However, for some tasks, standard edit similarities may satisfy the definition of (ϵ, γ, τ)-goodness poorly. Furthermore, existing methods for learning edit similarities rely on maximum likelihood and may not lead to an improved similarity function from an (ϵ, γ, τ)-goodness point of view. Kar & Jain (2011) propose to automatically adapt the goodness criterion to the problem at hand. In the rest of this chapter, we take a different approach: we see the (ϵ, γ, τ)-goodness as a novel, theoretically well-founded criterion to optimize an edit similarity.

5.4 Learning (ϵ, γ, τ)-Good Edit Similarity Functions

In this section, we propose a novel convex programming approach based on the theory of Balcan et al. (2008b) to learn (ϵ, γ, τ)-good edit similarity functions from both positive and negative pairs without requiring a costly iterative procedure. We will see in Section 5.5 that this framework allows us to derive generalization bounds establishing the consistency of our method and a relationship between the learned similarities and the generalization error of the linear classifier using it.

We begin this section by introducing an exponential-based edit similarity function that can be optimized in a direct way. Then, we present our convex programming approach to the problem of learning (ϵ, γ, τ)-good edit similarity functions, followed by a discussion on building relevant training pairs in this context. Finally, we end this section by showing that our approach can be straightforwardly adapted to tree edit similarity learning.

5.4.1 An Exponential-based Edit Similarity Function

In order to avoid the drawbacks of using iterative approaches such as EM for edit similarity learning, we propose to define an edit similarity for which the edit script does not depend on the edit costs.

Let $\mathbf{C} \in \mathbb{R}_+^{(|\Sigma|+1)\times(|\Sigma|+1)}$ be the edit cost matrix and for any $\mathsf{x}, \mathsf{x}' \in \Sigma^*$, let $\#(\mathsf{x}, \mathsf{x}')$ be a $(|\Sigma|+1) \times (|\Sigma|+1)$ matrix whose elements $\#_{i,j}(\mathsf{x}, \mathsf{x}')$ correspond to the number of times each edit operation $i \to j$ is used to turn x into x' in the Levenshtein script, $0 \leq i, j \leq |\Sigma|$. We define the following edit function:

$$e_{\mathbf{C}}(\mathsf{x}, \mathsf{x}') = \sum_{0 \leq i,j \leq |\Sigma|} C_{i,j} \#_{i,j}(\mathsf{x}, \mathsf{x}').$$

To compute $e_{\mathbf{C}}$, we do not extract the optimal script with respect to $\mathbf{C}$: we use the Levenshtein script[10] and apply custom costs $\mathbf{C}$ to it. Therefore, since the edit script defined by $\#(\mathsf{x}, \mathsf{x}')$ is fixed, $e_{\mathbf{C}}(\mathsf{x}, \mathsf{x}')$ is nothing more than a closed-form linear function of the edit costs and can be optimized directly.

Recall that a similarity function is assumed to be in $[-1, 1]$. To respect this requirement, we define our similarity function to be:

$$K_{\mathbf{C}}(\mathsf{x}, \mathsf{x}') = 2e^{-e_{\mathbf{C}}(\mathsf{x}, \mathsf{x}')} - 1.$$

Beyond this normalization requirement, the motivation for this exponential form is related to the one for using exponential kernels in SVM classifiers: it can be seen as a way

[10] In practice, one could use another type of script. We picked the Levenshtein script because it is a "reasonable" edit script, since it corresponds to a shortest script transforming x into x'.

to introduce nonlinearity to further separate examples of opposite class while moving closer those of the same class. Note that $K_{\mathbf{C}}$ may not be PSD nor symmetric. However, as we have seen earlier and unlike kernel theory, the theory of Balcan et al. (2008b) does not require these properties. This allows us to consider a broader type of edit similarity functions.

5.4.2 Learning the Edit Costs

We aim at learning the edit cost matrix $\mathbf{C}$ so as to optimize the (ϵ, γ, τ)-goodness of $K_{\mathbf{C}}$. We first focus on optimizing the goodness based on a relaxation of Definition 5.3, leading to a formulation based on the hinge loss (GESL$_{HL}$). Then, we introduce a more general version that can accommodate other loss functions (GESL$_L$).

5.4.2.1 Hinge Loss Formulation

Here, we want to learn $K_{\mathbf{C}}$ so that its hinge loss-based goodness (Definition 5.3) is optimized. More precisely, given a set of reasonable points and a margin γ, we want to optimize the amount of margin violation ϵ. Ideally, we would like to directly optimize Definition 5.3. Unfortunately, this would result in a nonconvex formulation (summing and subtracting up exponential terms) subject to local minima. Instead, we propose to optimize the following criterion:

$$\mathbb{E}_{(\mathsf{x},y)}\left[\mathbb{E}_{(\mathsf{x}',y')}\left[\left[1 - yy'K_{\mathbf{C}}(\mathsf{x},\mathsf{x}')/\gamma\right]_+ | R(\mathsf{x}')\right]\right] \leq \epsilon'. \tag{5.3}$$

Criterion (5.3) bounds that of Definition 5.3 due to the convexity of the hinge loss: clearly, if $K_{\mathbf{C}}$ satisfies (5.3), then it is (ϵ, γ, τ)-good in hinge loss with $\epsilon \leq \epsilon'$. Indeed, it is harder to satisfy since the "goodness" is required with respect to each reasonable point instead of considering the average similarity to these points. Therefore, optimizing $K_{\mathbf{C}}$ according to (5.3) implies the use of pair-based constraints.

Let us now consider a training sample $\mathcal{T} = \{z_i = (\mathsf{x_i}, y_i)\}_{i=1}^{n_{\mathcal{T}}}$ of $n_{\mathcal{T}}$ labeled instances. Recall that we do not know the set of reasonable points at this stage: they are inferred while learning the separator, that is, after the similarity is learned. For this reason, as in most metric learning methods, we will suppose that we are given pairs of examples. Formally, we suppose the existence of an indicator pairing function $f_{land} : \mathcal{T} \times \mathcal{T} \to \{0, 1\}$ which takes as input two training examples in $\mathcal{T}$ and returns 1 if they are paired and 0 otherwise. We assume that f_{land} associates to each element $z \in \mathcal{T}$ exactly $n_{\mathcal{L}}$ examples (called, with a slight abuse of language, the landmarks for z), leading to a total of $n_{\mathcal{T}} n_{\mathcal{L}}$ pairs. We discuss this matter further in Section 5.4.3.

Our formulation aims at fulfilling (5.3) for each (z_i, z_j) such that $f_{land}(z_i, z_j) = 1$. Therefore, we want $[1 - y_i y_j K_{\mathbf{C}}(\mathsf{x_i}, \mathsf{x_j})/\gamma]_+ = 0$, hence $y_i y_j K_{\mathbf{C}}(\mathsf{x_i}, \mathsf{x_j}) \geq \gamma$. A benefit

from using this constraint is that it can easily be turned into an equivalent linear one, considering the following two cases.

1. If $y_i \neq y_j$, we get:

$$-K_{\mathbf{C}}(\mathsf{x_i}, \mathsf{x_j}) \geq \gamma \iff e^{-e_{\mathbf{C}}(\mathsf{x_i},\mathsf{x_j})} \leq \frac{1-\gamma}{2} \iff e_{\mathbf{C}}(\mathsf{x_i}, \mathsf{x_j}) \geq -\log(\frac{1-\gamma}{2}).$$

   We can use a variable $B_1 \geq 0$ and write the constraint as $e_{\mathbf{C}}(\mathsf{x_i}, \mathsf{x_j}) \geq B_1$, with the interpretation that $B_1 = -\log(\frac{1-\gamma}{2})$. In fact, $B_1 \geq -\log(\frac{1}{2})$.

2. Likewise, if $y_i = y_j$, we get $e_{\mathbf{C}}(\mathsf{x_i}, \mathsf{x_j}) \leq -\log(\frac{1+\gamma}{2})$. We can use a variable $B_2 \geq 0$ and write the constraint as $e_{\mathbf{C}}(\mathsf{x_i}, \mathsf{x_j}) \leq B_2$, with the interpretation that $B_2 = -\log(\frac{1+\gamma}{2})$. In fact, $B_2 \in \left[0, -\log(\frac{1}{2})\right]$.

The optimization problem GESL$_{HL}$ can then be expressed as follows:

$$\begin{aligned}
(\text{GESL}_{HL}) \qquad & \min_{\mathbf{C}, B_1, B_2} \quad \frac{1}{n_{\mathcal{T}} n_{\mathcal{L}}} \sum_{\substack{1 \leq i \leq n_{\mathcal{T}}, \\ j: f_{land}(z_i, z_j)=1}} \ell_{HL}(\mathbf{C}, z_i, z_j) + \beta \|\mathbf{C}\|_{\mathcal{F}}^2 \\
& \text{s.t.} \quad B_1 \geq -\log(\tfrac{1}{2}), \quad 0 \leq B_2 \leq -\log(\tfrac{1}{2}), \quad B_1 - B_2 = \eta_\gamma \\
& \qquad C_{i,j} \geq 0, \quad 0 \leq i, j \leq |\Sigma|,
\end{aligned}$$

where $\beta \geq 0$ is a regularization parameter on edit costs, $\eta_\gamma \geq 0$ a parameter corresponding to the desired "margin" and

$$\ell_{HL}(\mathbf{C}, z_i, z_j) = \begin{cases} [B1 - e_{\mathbf{C}}(\mathsf{x_i}, \mathsf{x_j})]_+ & \text{if } y_i \neq y_j \\ [e_{\mathbf{C}}(\mathsf{x_i}, \mathsf{x_j}) - B2]_+ & \text{if } y_i = y_j \end{cases}.$$

The relationship between the margin γ and η_γ is given by $\gamma = \frac{e^{\eta_\gamma}-1}{e^{\eta_\gamma}+1}$. We chose Frobenius norm regularization because (i) it is simple, smooth and thus easier to optimize, and (ii) it allows us to derive generalization guarantees using uniform stability, as we will see in Section 5.5.

GESL$_{HL}$ is a convex program, thus one can efficiently find its global optimum. Using $n_{\mathcal{T}} n_{\mathcal{L}}$ slack variables to express each hinge loss, it has $O(n_{\mathcal{T}} n_{\mathcal{L}} + |\Sigma|^2)$ variables and $O(n_{\mathcal{T}} n_{\mathcal{L}})$ constraints. Note that GESL$_{HL}$ is a sparse convex program: each constraint involves at most one string pair and a limited number of edit cost variables, making the problem faster to solve. It is also worth noting that our approach is very flexible. First, it is general enough to be used with any definition of $e_{\mathbf{C}}$ that is based on an edit script (or even a convex combination of edit scripts). Second, one can incorporate additional convex constraints, for instance to include background knowledge or desired requirements on $\mathbf{C}$ (e.g., symmetry). Third, it can be easily adapted to the multi-class

case. Finally, it can be generalized to a larger class of loss functions, as we show in the following section.

5.4.2.2 General Formulation

In the previous section, we made use of the hinge loss-based Definition 5.3 to propose GESL_{HL}. Yet, other reformulations of Definition 5.1 are possible using any convex loss function that can be used to efficiently penalize the amount of violation ϵ with respect to margin γ. For instance, the logistic loss or the exponential loss could be used. This would also allow the derivation of learning guarantees (similar to Theorem 5.4) and an efficient learning rule.

Therefore, it is useful to be able to optimize a definition of (ϵ, γ, τ)-goodness based on a loss other than the hinge. Let $\ell(\mathbf{C}, z, z')$ be a convex loss function with respect to an edit cost matrix $\mathbf{C}$ and a pair of examples (z, z'). Our optimization problem can then be expressed in a more general form as follows:

$$(\text{GESL}_L) \qquad \min_{\mathbf{C}} \ \frac{1}{n_{\mathcal{T}} n_{\mathcal{L}}} \sum_{\substack{1 \leq i \leq n_{\mathcal{T}}, \\ j: f_{land}(z_i, z_j)=1}} \ell(\mathbf{C}, z_i, z_j) + \beta \|\mathbf{C}\|_{\mathcal{F}}^2.$$

In the rest of the paper, we will use GESL to refer to our approach in general, GESL_L when using an arbitrary loss function ℓ and GESL_{HL} for the specific case of the hinge loss.

5.4.3 Pairing strategy

The question of how one should define the pairing function f_{land} relates to the open question of building training pairs in many metric learning problems. In some applications, the answer may be trivial: for instance, a misspelled word and its correction. Otherwise, popular choices are to pair each example with its nearest neighbor, random pairing or simply to consider all possible pairs.

On the other hand, the (ϵ, γ, τ)-goodness of the similarity should be improved with respect to the reasonable points, a subset of examples of probability τ that allows low error and large margin. However, this set depends on the similarity function itself and is thus unknown beforehand. Yet, a relevant strategy in the context of (ϵ, γ, τ)-goodness may be to improve the similarity with respect to carefully selected examples rather than considering all possible pairs. Consequently, we consider two pairing strategies that will be compared in our experiments (Section 5.6):

1. *Levenshtein pairing*: we pair each $z \in \mathcal{T}$ with its $n_{\mathcal{T}}$ nearest neighbors of the same class and its $n_{\mathcal{T}}$ farthest neighbors of the opposite class, using the Levenshtein distance. This pairing strategy is meant to capture the essence of Definition 5.1 and in particular the idea that reasonable points "represent" the data well. Essentially, we pair z with a few points that are already good representatives of z and optimize the edit costs so that they become even better representatives. Note that the choice of the Levenshtein distance to pair examples is consistent with our choice to define $e_{\mathbf{C}}$ according to the Levenshtein script.

2. *Random pairing*: we pair each $z \in \mathcal{T}$ with a number $n_{\mathcal{T}}$ of randomly chosen examples of the same class and $n_{\mathcal{T}}$ randomly chosen examples of the opposite class.

In either case, we have $n_{\mathcal{L}} = 2N = \alpha n_{\mathcal{T}}$ with $0 < \alpha \leq 1$. Taking $\alpha = 1$ corresponds to considering all possible pairs. In a sense, α can be seen as playing the role of τ (which gives the proportion of points that are reasonable in the definition of goodness) at the pair level, even though no direct relation can be made between the two.

5.4.4 Adaptation to trees

So far, we have implicitly considered that the data are strings. In this section, before presenting a theoretical analysis of GESL, we show that it may be used in a simple and efficient way to learn tree edit similarities. As mentioned in Section 5.4.1, our edit function, defined as

$$e_{\mathbf{C}}(\mathsf{x}, \mathsf{x}') = \sum_{0 \leq i,j \leq |\Sigma|} C_{i,j} \#_{i,j}(\mathsf{x}, \mathsf{x}'),$$

is nothing more than a linear combination of the edit costs, where $\#_{i,j}(\mathsf{x}, \mathsf{x}')$ is the number of times the edit operation $i \to j$ occurs in the Levenshstein script turning x into x'. This opens the door to a straightforward generalization of GESL to tree edit distance: instead of a string edit script, we can use a tree edit script according to either variant of the tree edit distance (Zhang & Shasha, 1989; Selkow, 1977) and solve the (otherwise unchanged) optimization problem presented in Section 5.4.2. This allows us, once again, to avoid using a costly iterative procedure. We only have to compute the edit script between two trees once, which dramatically reduces the algorithmic complexity of the learning algorithm. Moreover, we will see that the theoretical analysis of GESL presented in the following section holds for tree edit similarity learning.

5.5 Theoretical Analysis

This section presents a theoretical analysis of GESL. In Section 5.5.1, we derive a generalization bound guaranteeing its consistency and relating to the (ϵ, γ, τ)-goodness in

generalization of the learned similarity function, and thus to the true risk of the linear classifier. This theoretical study is performed for a large class of loss functions. In Section 5.5.2, we instantiate this generalization bound for the specific case of the hinge loss (GESL_{HL}). Finally, Section 5.5.3 is devoted to a discussion about the main features of the bounds, and to the presentation of a way to get rid of the assumption that the length of the strings (or the size of the trees) is bounded.

5.5.1 Generalization Bound for General Loss Functions

As pointed out in Chapter 3, the training pairs used in metric learning are not i.i.d. and therefore the classic results of statistical learning theory do not directly hold. To derive a generalization bound for GESL_L, we build upon the adaptation of uniform stability to the metric learning case (Jin et al., 2009) and extend it to edit similarity learning. We first prove that GESL_L has a uniform stability: this is established in Theorem 5.9, using Lemma 5.8 and the assumption of k-lipschitzness (Definition 5.5). The stability property allows us to derive our generalization bound (Theorem 5.13) using the McDiarmid inequality (Theorem 5.10) and the assumption of (σ, m)-admissibility (Definition 5.6).

We denote the objective function of GESL_L by:

$$F_{\mathcal{T}}(\mathbf{C}) = \frac{1}{n_{\mathcal{T}}} \sum_{k=1}^{n_{\mathcal{T}}} \frac{1}{n_{\mathcal{L}}} \sum_{j=1}^{n_{\mathcal{L}}} \ell(\mathbf{C}, z_k, z'_{k_j}) + \beta \|\mathbf{C}\|_{\mathcal{F}}^2,$$

where z'_{k_j} denotes the jth landmark associated to z_k and $\ell(\mathbf{C}, z_k, z'_{k_j})$ the loss for a pair of examples with respect to an edit cost matrix $\mathbf{C}$.

The first term of $F_{\mathcal{T}}(\mathbf{C})$ is the empirical risk $R^{\ell}_{\mathcal{T}}(\mathbf{C})$ over the training sample $\mathcal{T}$. The true risk $R^{\ell}(\mathbf{C})$ is given by:

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

Recall that our empirical risk is not defined over all possible training pairs, unlike most metric learning algorithms, but according to some particular landmark examples. On the other hand, the true risk is defined over any pair of instances. For notational convenience, we also introduce the estimation error $D_{\mathcal{T}}$, which is the deviation between the true risk and the empirical risk:

$$D_{\mathcal{T}} = R^{\ell}(\mathbf{C}_{\mathcal{T}}) - R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}}),$$

where $\mathbf{C}_{\mathcal{T}}$ denotes the edit cost matrix learned by GESL_L from $\mathcal{T}$.

In this section, we propose an analysis that holds for a large class of loss functions. We consider loss functions ℓ that fulfill the k-lipschitz property with respect to the first argument $\mathbf{C}$ (Definition 5.5) and the definition of (σ, m)-admissibility (Definition 5.6).

Definition 5.5. A loss function $\ell(\mathbf{C}, z_1, z_2)$ is k-lipschitz with respect to its first argument if for any matrices $\mathbf{C}, \mathbf{C}'$ and any pair of labeled examples (z_1, z_2):

$$|\ell(\mathbf{C}, z_1, z_2) - \ell(\mathbf{C}', z_1, z_2)| \leq k\|\mathbf{C} - \mathbf{C}'\|_{\mathcal{F}}.$$

Definition 5.6. A loss function $\ell(\mathbf{C}, z_1, z_2)$ is (σ, m)-*admissible*, with respect to $\mathbf{C}$, if (i) it is convex with respect to its first argument and (ii) the following condition holds:

$$\forall z_1, z_2, z_3, z_4, |\ell(\mathbf{C}, z_1, z_2) - \ell(\mathbf{C}, z_3, z_4)| \leq \sigma|y_1 y_2 - y_3 y_4| + m$$

with $z_i = (x_i, y_i)$, for $i = 1, 2, 3, 4$, are labeled examples.

Definition 5.6 requires the deviation of the losses between two pairs of examples to be bounded by a value that depends only on the labels and on some constants independent from the examples and the cost matrix $\mathbf{C}$. It follows that the labels must be bounded, which is not a strong assumption in the classification setting we are interesting in. In our case, we have binary labels ($y_i \in \{-1, 1\}$), which implies that the quantity $|y_1 y_2 - y_3 y_4|$ is either 0 or 2. We will see in Section 5.5.2 that the hinge loss of GESL$_{HL}$ satisfies Definition 5.5 and Definition 5.6. This can also be shown for other popular loss functions, such as the logistic loss or the exponential loss.[11]

Note that from the convexity of ℓ with respect to its first argument, it follows that R^{ℓ}, $R^{\ell}_{\mathcal{T}}$ and $F_{\mathcal{T}}$ are convex functions.

Our objective is to derive an upper bound on the true risk $R^{\ell}(C_{\mathcal{T}})$ with respect to the empirical risk $R^{\ell}_{\mathcal{T}}(C_{\mathcal{T}})$ using uniform stability (Definition 2.8) adapted to the case where training data consist of pairs (Jin et al., 2009).

Definition 5.7 (Jin et al., 2009)**.** A learning algorithm has a uniform stability in $\frac{\kappa}{n_{\mathcal{T}}}$, where κ is a positive constant, if

$$\forall(\mathcal{T}, z), \forall i, \sup_{z_1, z_2} |\ell(\mathbf{C}_{\mathcal{T}}, z_1, z_2) - \ell(\mathbf{C}_{\mathcal{T}^{i,z}}, z_1, z_2)| \leq \frac{\kappa}{n_{\mathcal{T}}},$$

where $\mathcal{T}^{i,z}$ is the new set obtained by replacing $z_i \in \mathcal{T}$ by a new example z.

To prove that GESL$_L$ has the property of uniform stability, we need the following lemma and the k-lipschitz property of ℓ.

Lemma 5.8. *Let $F_{\mathcal{T}}$ and $F_{\mathcal{T}^{i,z}}$ be the functions to optimize, $\mathbf{C}_{\mathcal{T}}$ and $\mathbf{C}_{\mathcal{T}^{i,z}}$ their corresponding minimizers, and β the regularization parameter used in GESL$_L$. Let*

[11] To satisfy Definition 5.5, their domain must be bounded (Rosasco et al., 2004).

$\Delta\mathbf{C} = (\mathbf{C}_{\mathcal{T}} - \mathbf{C}_{\mathcal{T}^{i,z}})$. *For any* $t \in [0,1]$*:*

$$\|\mathbf{C}_{\mathcal{T}}\|_{\mathcal{F}}^2 - \|\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}\|_{\mathcal{F}}^2 + \|\mathbf{C}_{\mathcal{T}^{i,z}}\|_{\mathcal{F}}^2 - \|\mathbf{C}_{\mathcal{T}^{i,z}} + t\Delta\mathbf{C}\|_{\mathcal{F}}^2 \leq \frac{(2n_{\mathcal{T}} + n_{\mathcal{L}})t2k}{\beta n_{\mathcal{T}} n_{\mathcal{L}}}\|\Delta\mathbf{C}\|_{\mathcal{F}}.$$

Proof. See Appendix B.1.1. □

We can now prove the stability of GESL$_L$.

Theorem 5.9 (Stability of **GESL$_\mathbf{L}$**)**.** *Let* $n_{\mathcal{T}}$ *and* $n_{\mathcal{L}}$ *be respectively the number of training examples and landmark points. Assuming that* $n_{\mathcal{L}} = \alpha n_{\mathcal{T}}$*,* $\alpha \in]0,1]$*, and that the loss function used in GESL$_L$ is* k*-lipschitz, then GESL$_L$ has a uniform stability in* $\frac{\kappa}{n_{\mathcal{T}}}$*, where* $\kappa = \frac{2(2+\alpha)k^2}{\beta\alpha}$*.*

Proof. Using $t = 1/2$ on the left-hand side of Lemma 5.8, we get

$$\|\mathbf{C}_{\mathcal{T}}\|_{\mathcal{F}}^2 - \|\mathbf{C}_{\mathcal{T}} - \frac{1}{2}\Delta\mathbf{C}\|_{\mathcal{F}}^2 + \|\mathbf{C}_{\mathcal{T}^{i,z}}\|_{\mathcal{F}}^2 - \|\mathbf{C}_{\mathcal{T}^{i,z}} + \frac{1}{2}\Delta\mathbf{C}\|_{\mathcal{F}}^2 = \frac{1}{2}\|\Delta\mathbf{C}\|_{\mathcal{F}}^2.$$

Then, applying Lemma 5.8, we get

$$\frac{1}{2}\|\Delta\mathbf{C}\|_{\mathcal{F}}^2 \leq \frac{(2n_{\mathcal{T}} + n_{\mathcal{L}})k}{\beta n_{\mathcal{T}} n_{\mathcal{L}}}\|\Delta\mathbf{C}\|_{\mathcal{F}} \Rightarrow \|\Delta\mathbf{C}\|_{\mathcal{F}} \leq \frac{2(2n_{\mathcal{T}} + n_{\mathcal{L}})k}{\beta n_{\mathcal{T}} n_{\mathcal{L}}}.$$

Now, from the k-lipschitz property of ℓ, we have for any z, z'

$$|\ell(\mathbf{C}_{\mathcal{T}}, z, z') - \ell(\mathbf{C}_{\mathcal{T}^{i,z}}, z, z')| \leq k\|\Delta\mathbf{C}\|_{\mathcal{F}} \leq \frac{2(2n_{\mathcal{T}} + n_{\mathcal{L}})k^2}{\beta n_{\mathcal{T}} n_{\mathcal{L}}}.$$

Replacing $n_{\mathcal{L}}$ by $\alpha n_{\mathcal{T}}$ completes the proof. □

Now, using the property of stability, we can derive our generalization bound over $R^{\ell}(\mathbf{C}_{\mathcal{T}})$. This is done by using the McDiarmid inequality (McDiarmid, 1989).

Theorem 5.10 (McDiarmid inequality)**.** *Let* $X_1, \ldots, X_n$ *be* n *independent random variables taking values in* X *and let* $Z = f(X_1, \ldots, X_n)$*. If for each* $1 \leq i \leq n$*, there exists a constant* c_i *such that*

$$\sup_{x_1,\ldots,x_n,x_i' \in \mathcal{X}} |f(x_1, \ldots, x_n) - f(x_1, \ldots, x_i', \ldots, x_n)| \leq c_i, \forall 1 \leq i \leq n,$$

$$\textit{then for any } \epsilon > 0, \qquad \Pr[|Z - \mathbb{E}[Z]| \geq \epsilon] \leq 2\exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}\right).$$

To derive our bound on $R^{\ell}(\mathbf{C}_{\mathcal{T}})$, we just need to replace Z by $D_{\mathcal{T}}$ in Theorem 5.10 and to bound $\mathbb{E}_{\mathcal{T}}[D_{\mathcal{T}}]$ and $|D_{\mathcal{T}} - D_{\mathcal{T}^{i,z}}|$, which is shown by the following lemmas.

Lemma 5.11. *For any learning method of estimation error* $D_{\mathcal{T}}$ *and satisfying a uniform stability in* $\frac{\kappa}{n_{\mathcal{T}}}$*, we have* $\mathbb{E}_{\mathcal{T}}[D_{\mathcal{T}}] \leq \frac{2\kappa}{n_{\mathcal{T}}}$*.*

Proof. See Appendix B.1.2. □

Lemma 5.12. *For any edit cost matrix learned by GESL$_L$ using $n_\mathcal{T}$ training examples and $n_\mathcal{L}$ landmarks, and any loss function ℓ satisfying (σ, m)-admissibility, we have the following bound:*

$$\forall i, 1 \leq i \leq n_\mathcal{T}, \quad \forall z, \quad |D_\mathcal{T} - D_{\mathcal{T}^{i,z}}| \leq \frac{2\kappa}{n_\mathcal{T}} + \frac{(2n_\mathcal{T} + n_\mathcal{L})(2\sigma + m)}{n_\mathcal{T} n_\mathcal{L}}.$$

Proof. See Appendix B.1.3. □

We are now able to derive our generalization bound over $R^\ell(\mathbf{C}_\mathcal{T})$.

Theorem 5.13 (Generalization bound for GESL$_L$)**.** *Let $\mathcal{T}$ be a sample of $n_\mathcal{T}$ randomly selected training examples and let $\mathbf{C}_\mathcal{T}$ be the edit cost matrix learned by GESL$_L$ with stability $\frac{\kappa}{n_\mathcal{T}}$. Assuming that $\ell(\mathbf{C}_\mathcal{T}, z, z')$ is k-lipschitz and (σ, m)-admissible, and using $n_\mathcal{L} = \alpha n_\mathcal{T}$ landmark points, with probability $1 - \delta$, we have the following bound for $R^\ell(\mathbf{C}_\mathcal{T})$:*

$$R^\ell(\mathbf{C}_\mathcal{T}) \leq R^\ell_\mathcal{T}(\mathbf{C}_\mathcal{T}) + 2\frac{\kappa}{n_\mathcal{T}} + \left(2\kappa + \frac{2+\alpha}{\alpha}(2\sigma + m)\right)\sqrt{\frac{\ln(2/\delta)}{2n_\mathcal{T}}}$$

with $\kappa = \frac{2(2+\alpha)k^2}{\alpha\beta}$.

Proof. Recall that $D_\mathcal{T} = R^\ell(\mathbf{C}_\mathcal{T}) - R^\ell_\mathcal{T}(\mathbf{C}_\mathcal{T})$ and $n_\mathcal{L} = \alpha n_\mathcal{T}$. From Lemma 5.12, we get

$$|D_\mathcal{T} - D_{\mathcal{T}^{i,z}}| \leq \sup_{\mathcal{T}, z'} |D_\mathcal{T} - D_{\mathcal{T}^{i,z'}}| \leq \frac{2\kappa + B}{n_\mathcal{T}} \text{ with } B = \frac{(2+\alpha)}{\alpha}(2\sigma + m).$$

Then by applying the McDiarmid inequality, we have

$$\Pr[|D_\mathcal{T} - \mathbb{E}_T[D_\mathcal{T}]| \geq \epsilon] \leq 2\exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{n_\mathcal{T}} \frac{(2\kappa+B)^2}{n_\mathcal{T}^2}}\right) \leq 2\exp\left(-\frac{2\epsilon^2}{\frac{(2\kappa+B)^2}{n_\mathcal{T}}}\right). \tag{5.4}$$

By fixing $\delta = 2\exp\left(-\frac{2\epsilon^2}{(2\kappa+B)^2/n_\mathcal{T}}\right)$, we get $\epsilon = (2\kappa + B)\sqrt{\frac{\ln(2/\delta)}{2n_\mathcal{T}}}$. Finally, from (5.4), Lemma 5.11 and the definition of $D_\mathcal{T}$, we have with probability at least $1 - \delta$:

$$D_\mathcal{T} < \mathbb{E}_T[D_\mathcal{T}] + \epsilon \Rightarrow R^\ell(\mathbf{C}_\mathcal{T}) < R^\ell_\mathcal{T}(\mathbf{C}_\mathcal{T}) + 2\frac{\kappa}{n_\mathcal{T}} + (2\kappa + B)\sqrt{\frac{\ln(2/\delta)}{2n_\mathcal{T}}},$$

which gives the theorem. □

5.5.2 Generalization Bound for the Hinge Loss

Theorem 5.13 holds for any loss function $\ell(\mathbf{C}_\mathcal{T}, z, z')$ that is k-lipschitz and (σ, m)-admissible with respect to $\mathbf{C}_\mathcal{T}$. Let us now rewrite this bound when ℓ is the hinge loss-based function ℓ_{HL} used in GESL$_{HL}$. We first have to prove that ℓ_{HL} is k-lipschitz (Lemma 5.14) and (σ, m)-admissible (Lemma 5.16). Then, we derive the generalization bound for GESL$_{HL}$.

In order to fulfill the k-lipschitz and (σ, m)-admissibility properties, we suppose every string length bounded by a constant $W > 0$. Since the Levenshtein script between two strings x and x' contains at most $\max(|\mathsf{x}|, |\mathsf{x}'|)$ operations, we have

$$\|\#(\mathsf{x}, \mathsf{x}')\|_\mathcal{F} = \sqrt{\sum_{l,c} \#_{l,c}(\mathsf{x}, \mathsf{x}')^2} \leq \sqrt{\left(\sum_{l,c} \#_{l,c}(\mathsf{x}, \mathsf{x}')\right)^2} \leq W.$$

When dealing with labeled instances, we will sometimes denote $\|\#(\mathsf{x}, \mathsf{x}')\|_\mathcal{F} \leq W$ by $\|\#(z_1, z_2)\|_\mathcal{F} \leq W$ for the sake of convenience.

Lemma 5.14. *The function ℓ_{HL} is k-lipschitz with $k = W$.*

Proof. See Appendix B.1.4. □

We will now prove that ℓ_{HL} is (σ, m)-admissible for any optimal solution $\mathbf{C}_\mathcal{T}$ learned by GESL$_{HL}$ (Lemma 5.16). To be able to do this, we must show that the norm of $\mathbf{C}_\mathcal{T}$ is bounded (Lemma 5.15).

Lemma 5.15. *Let $(\mathbf{C}_\mathcal{T}, B_1, B_2)$ an optimal solution learned by GESL$_{HL}$ from a training sample $\mathcal{T}$, and let $B_\gamma = max(\eta_\gamma, -log(1/2))$. Then $\|\mathbf{C}_\mathcal{T}\|_\mathcal{F} \leq \sqrt{\frac{B_\gamma}{\beta}}$.*

Proof. See Appendix B.1.5. □

Lemma 5.16. *For any optimal solution $(\mathbf{C}_\mathcal{T}, B_1, B_2)$, ℓ_{HL} is (σ, m)-admissible with $\sigma = \frac{\sqrt{\frac{B_\gamma}{\beta}}W + 3B_\gamma}{2}$ and $m = \sqrt{\frac{B_\gamma}{\beta}}W$, with $B_\gamma = \max(\eta_\gamma, -\log(1/2))$.*

Proof. Let $\mathbf{C}_\mathcal{T}$ be an optimal solution learned by GESL$_{HL}$ from a training sample T and let z_1, z_2, z_3, z_4 be four labeled examples. We study two cases:

1. If $y_1 y_2 = y_3 y_4$, regardless of the label values, using the 1-lispschitz property of the hinge loss, B_1 (when $y_1 y_2 = y_3 y_4 = -1$) or B_2 ($y_1 y_2 = y_3 y_4 = 1$) cancels out (in a similar way as in Appendix 5.14) and thus :

$$\begin{aligned} |\ell_{HL}(\mathbf{C}_\mathcal{T}, z_1, z_2) - \ell_{HL}(\mathbf{C}_\mathcal{T}, z_3, z_4)| &\leq \|\mathbf{C}_\mathcal{T}\|_\mathcal{F} \|\#(z_1, z_2) - \#(z_3, z_4)\|_\mathcal{F} \\ &\leq \sqrt{\frac{B_\gamma}{\beta}} W \qquad \text{from Lemma 5.15.} \end{aligned}$$

2. Otherwise, if $y_1y_2 \neq y_3y_4$, note that $|B_1+B_2| = \eta_\gamma+2B_2 \leq 3B_\gamma$ and $|y_1y_2-y_3y_4| = 2$. Hence, whatever the labels of the examples compatible with this case, by using the 1-lipschitz property of hinge loss and application of the triangular inequality, we get

$$\begin{aligned}
|\ell_{HL}(\mathbf{C}_\mathcal{T}, z_1, z_2) - \ell_{HL}(\mathbf{C}_\mathcal{T}, z_3, z_4)| &\leq |\sum_{l,c} C_{\mathcal{T},l,c}(\#_{l,c}(z_1,z_2) + \#_{l,c}(z_3,z_4))| + \\
&\quad |B_1 + B_2| \\
&\leq \|\mathbf{C}_\mathcal{T}\|_\mathcal{F} \|\#(z_1,z_2) + \#(z_3,z_4)\|_\mathcal{F} + 3B_\gamma \\
&\leq \sqrt{\frac{B_\gamma}{\beta}} 2W + 3B_\gamma \\
&\leq \frac{\sqrt{\frac{B_\gamma}{\beta}}W + 3B_\gamma}{2} |y_1y_2 - y_3y_4| + \sqrt{\frac{B_\gamma}{\beta}} W.
\end{aligned}$$

Then, by choosing $\sigma = \frac{\sqrt{\frac{B_\gamma}{\beta}}W+3B_\gamma}{2}$ and $m = \sqrt{\frac{B_\gamma}{\beta}}W$, we have that ℓ_{HL} is (σ, m)-admissible. □

We can now give the convergence bound for GESL$_{HL}$.

Theorem 5.17 (Generalization bound for GESL$_{HL}$)**.** *Let $\mathcal{T}$ be a sample of $n_\mathcal{T}$ randomly selected training examples and let $\mathbf{C}_\mathcal{T}$ be the edit cost matrix learned by GESL$_{HL}$ with stability $\frac{\kappa}{n_\mathcal{T}}$ using $n_\mathcal{L} = \alpha n_\mathcal{T}$ landmark points. With probability $1-\delta$, we have the following bound for $R^\ell(\mathbf{C}_\mathcal{T})$:*

$$R^\ell(\mathbf{C}_\mathcal{T}) \leq R^\ell_\mathcal{T}(\mathbf{C}_\mathcal{T}) + 2\frac{\kappa}{n_\mathcal{T}} + \left(2\kappa + \frac{2+\alpha}{\alpha}\left(\frac{2W}{\sqrt{\beta B_\gamma}} + 3\right) B_\gamma\right)\sqrt{\frac{\ln(2/\delta)}{2n_\mathcal{T}}}$$

with $\kappa = \frac{2(2+\alpha)W^2}{\alpha\beta}$ and $B_\gamma = \max(\eta_\gamma, -log(1/2))$.

Proof. It directly follows from Theorem 5.13, Lemma 5.14 and Lemma 5.16 by noting that $2\sigma + m = \left(\frac{2W}{\sqrt{\beta B_\gamma}} + 3\right) B_\gamma$. □

5.5.3 Discussion

The generalization bounds presented in Theorem 5.13 and Theorem 5.17 outline three important features of our approach. To begin with, it has a classic $O(\sqrt{1/n_\mathcal{T}})$ convergence rate. Second, this rate of convergence is independent of the alphabet size, which means that our method should scale well to problems with large alphabets. We will see in Section 5.6 that it is actually the case in practice. Finally, thanks to the relation between the optimized criterion and the definition of (ϵ, γ, τ)-goodness that we established

earlier, these bounds also ensure the goodness in generalization of the learned similarity function. Therefore, they guarantee that the similarity will induce classifiers with small true risk for the classification task at hand.

Note that to derive Theorem 5.17, we assumed the size of the strings was bounded by a constant W. Even though this is not a strong restriction, it would be interesting to get rid of this assumption and derive a bound that is independent of W. This is possible when the marginal distribution of P over the set of strings follows a generative model ensuring that the probability of a string decreases exponentially fast with its length. In this case, we can use the fact that very long strings have a very small probability to occur. Then with high probability, we can bound the maximum string length in a sample and remove W from the generalization bound. Indeed, one can show that for any string stochastic language p defined by a probabilistic automaton (Denis et al., 2006) or a stochastic context-free grammar (Etessami & Yannakakis, 2009), there exist some constants $U > 0$ and $0 < \rho < 1$ such that the sum of the probabilities of strings of length at least k is bounded:

$$\sum_{x,|x|>=k} p(x) < U\rho^k. \tag{5.5}$$

To take into account this result in our framework, we need an estimation of the length of the examples used to derive the generalization bound, that is, a sample of $n_{\mathcal{T}}$ examples with two additional examples z and z'. For any sample of $n_{\mathcal{T}} + 2$ strings identically and independently drawn from p, we can bound the length of any string x of this sample. With a confidence greater than $1 - \delta/2(n_{\mathcal{T}} + 2)$, we have:

$$|x| < \frac{\log(U2(n_{\mathcal{T}} + 2)/\delta)}{\log(1/\rho)},$$

by fixing $\delta/2(n_{\mathcal{T}} + 2) = U\rho^k$.

Applying this result to every string of the sample, we get that with probability at least $1 - \delta/2$, any sample of $n_{\mathcal{T}} + 2$ elements has only strings of size at most $\frac{\log(U2(n_{\mathcal{T}}+2)/\delta)}{\log(1/\rho)}$. Then, by using Theorem 5.17 with a confidence $\delta/2$ and replacing W by $\frac{\log(2(n_{\mathcal{T}}+2)U/\delta)}{\log(1/\rho)}$, we obtain the following bound.

Theorem 5.18. *Let $\mathcal{T}$ be a sample of $n_{\mathcal{T}}$ randomly selected training examples drawn from a stochastic language p and let $\mathbf{C}_{\mathcal{T}}$ be the edit costs learned by GESL$_{HL}$ with stability $\frac{\kappa}{n_{\mathcal{T}}}$ using $n_{\mathcal{L}} = \alpha n_{\mathcal{T}}$ landmark points. Then there exists constants $U > 0$ and $0 < \rho < 1$ such that with probability at least $1 - \delta$, we have:*

$$R^{\ell}(\mathbf{C}_{\mathcal{T}}) \leq R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}}) + 2\frac{\kappa}{n_{\mathcal{T}}} + \left(2\kappa + \frac{2+\alpha}{\alpha}\left(\frac{2\log(2(n_{\mathcal{T}}+2)U/\delta)}{\sqrt{\beta B_\gamma}\log(1/\rho)} + 3\right)B_\gamma\right)\sqrt{\frac{\ln(4/\delta)}{2n_{\mathcal{T}}}}$$

with $\kappa = \frac{2(2+\alpha)\log^2(2(n_{\mathcal{T}}+2)U/\delta)}{\alpha\beta\log^2(1/\rho)}$ and $B_\gamma = \max(\eta_\gamma, -\log(1/2))$.

Finally, let us conclude this section by discussing the adaptation of the entire theoretical analysis to tree edit similarity learning. The generalization bound for GESL$_L$ (Theorem 5.13) holds for trees since the arguments used in Section 5.5.1 are not specific to strings. Regarding the bound for GESL$_{HL}$ (Theorem 5.17), we used the assumption that the length of the strings is bounded by a constant W. This can be easily adapted to trees: if we assume that the size of each tree (in its number of nodes) is bounded by W, Theorem 5.17 also holds. Finally, the arguments for deriving a bound independent of the constant W hold for trees since the property (5.5) is also valid for rational stochastic tree languages (Denis et al., 2008).

5.6 Experimental Validation

In this section, we provide an experimental evaluation of GESL$_{HL}$.[12] We are interested in evaluating the performance of different (standard or learned) edit similarities directly plugged into linear classifiers, as suggested by the theory of (ϵ, γ, τ)-goodness presented in Section 5.2. Linear classifiers are learned using Balcan's learning rule (5.2). We compare three edit similarity functions: (i) $K_{\mathbf{C}}$, learned by GESL$_{HL}$,[13] (ii) the Levenshtein distance d_{lev}, which constitutes the baseline, and (iii) an edit similarity function p_e learned with an EM-like algorithm (Oncina & Sebban, 2006). We show results on the same datasets as in the preliminary study (Section 5.3): English and French words (Section 5.6.1) and handwritten digits (Section 5.6.2).

5.6.1 English and French Words

Recall that the task is to learn a model to classify words as either English or French. We use the 2,000 top words lists from Wiktionary.[14]

5.6.1.1 Convergence rate

We first assess the convergence rate of the two considered edit cost learning methods (i and iii). We keep aside 600 words as a validation set to tune the parameters, using 5-fold cross-validation and selecting the value offering the best classification accuracy. We then build bootstrap samples $\mathcal{T}$ from the remaining 2,000 words to learn the edit

[12] An open-source implementation of our method is available at: http://labh-curien.univ-st-etienne.fr/~bellet/.

[13] In this series of experiments, we constrained the cost matrices to be symmetric to be independent from the order in which the instances are paired.

[14] These lists are available at http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists. We only considered unique words (i.e., not appearing in both lists) of length at least 4, and we also got rid of accent and punctuation marks. We ended up with about 2,600 words over an alphabet of 26 symbols.

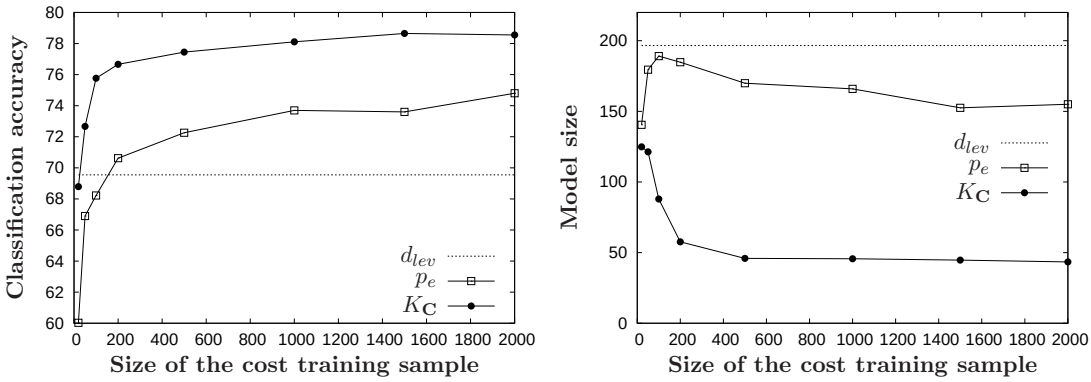


FIGURE 5.8: Learning the edit costs: accuracy and sparsity results (Word dataset).

costs (5 runs for each size $n_{\mathcal{T}}$), as well as 600 words to train the separator $\boldsymbol{\alpha}$ and 400 words to test its performance.

Figure 5.8 shows the accuracy and sparsity results of each method with respect to $n_{\mathcal{T}}$, averaged over 5 runs. We see that $K_{\mathbf{C}}$ leads to more accurate classifiers than d_{lev} and p_e for $n_{\mathcal{T}} > 20$. The difference is statistically significant: the Student's t-test yields a p-value < 0.01. At the same time, $K_{\mathbf{C}}$ requires 3 to 4 times less reasonable points, thus increasing classification speed by just as much. The exact figures are as follows: d_{lev} achieves 69.55% accuracy with a model size of 197, p_e achieves at best 74.80% with a model size of 155, and $K_{\mathbf{C}}$ achieves at best 78.65% with a model size of only 45. This clearly indicates that GESL$_{HL}$ leads to a better similarity than (ii) and (iii). Moreover, the convergence rate of GESL$_{HL}$ is very fast, considering that $(26+1)^2 = 729$ costs must be learned: it needs very few examples (about 20) to outperform the Levenshtein distance, and about 200 examples to reach convergence. This provides experimental evidence that our method scales well with the size of the alphabet, as suggested by the generalization bound derived in Section 5.5.2. On the other hand, (iii) seems to suffer from the large number of costs to estimate: it needs a lot more examples to outperform Levenshtein (about 200) and convergence seems to be only reached at 1,000.

5.6.1.2 Pairing strategy and influence of α

In the previous experiment, the pairing strategy and the value of α was set by cross-validation. In this section, we compare the two pairing strategies (random pairing and Levenshtein pairing) presented in Section 5.4.3 as well as the influence of α (the proportion of landmarks associated with each training example). Figure 5.9 shows the accuracy and sparsity results obtained for $n_{\mathcal{T}} = 1,500$ with respect to α and the pairing strategies.[15] The accuracy for d_{lev} and p_e is carried over from Figure 5.8 for comparison

[15] We do not evaluate the pairing strategies on the whole data ($n_{\mathcal{T}} = 2,000$) so that we can build 5 bootstrap samples and average the results over these.

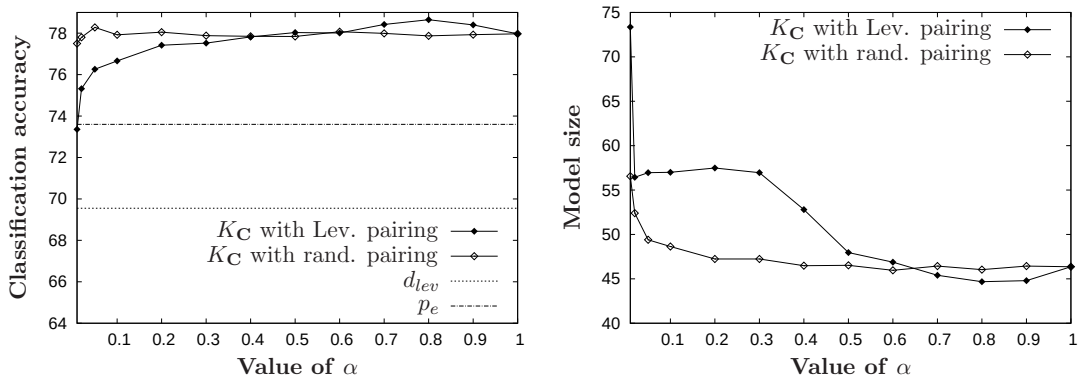


FIGURE 5.9: Pairing strategies: accuracy and sparsity results w.r.t. α (Word dataset).

(model sizes for d_{lev} and p_e, which are not shown for scale reasons, are 197 and 152 respectively).

These results are very informative. Regardless of the pairing strategy, $K_{\mathbf{C}}$ outperforms d_{lev} and p_e even when making use of a very small proportion of the available pairs (1%), which tremendously reduces the complexity of the similarity learning phase. Random pairing gives better results than Levenshtein pairing for $\alpha \leq 0.4$. When $\alpha \geq 0.6$, this trend is reversed. This means that for a small proportion of pairs, we learn better from pairing random landmarks than from pairing landmarks that are already good representatives of the training examples. On the other hand, when the proportion increases, Levenshtein pairing allows us to avoid pairing examples with the "worst" landmarks: best results are obtained with Levenshtein pairing and $\alpha = 0.8$.

5.6.1.3 Learning the separator

We now assess the performance of the three edit similarities with respect to the number of examples n used to learn the separator $\boldsymbol{\alpha}$. For $K_{\mathbf{C}}$ and p_e, we use the edit cost matrix that performed best in Section 5.6.1.1. Taking our set of 2,000 words, we keep aside 400 examples to test the models and build bootstrap samples from the remaining 1,600 words to learn $\boldsymbol{\alpha}$. Figure 5.10 shows the accuracy and sparsity results of each method with respect to n, averaged over 5 runs. Again, $K_{\mathbf{C}}$ outperforms d_{lev} and p_e for every size n (the difference is statistically significant with a p-value < 0.01 using a Student's t-test) while always leading to (up to 5 times) sparser models. Moreover, the size of the models induced by $K_{\mathbf{C}}$ stabilizes for $n \geq 400$ while the accuracy still increases. This is not the case for the models induced by d_{lev} and p_e, whose size keeps growing. To sum up, the best similarity learned by GESL$_{HL}$ outperforms the best similarity learned with the method of Oncina & Sebban (2006), which had been proven to outperform other state-of-the-art methods.

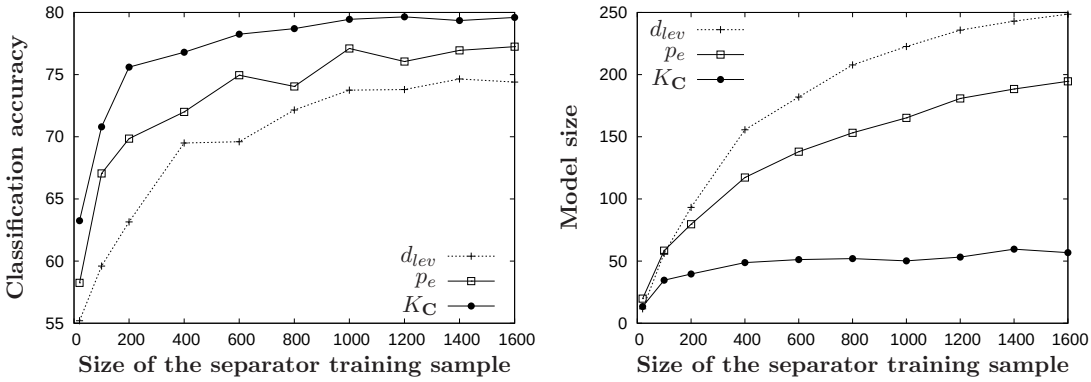


FIGURE 5.10: Learning the separator: accuracy and sparsity results (Word dataset).

English			French		
high	showed	holy	economiques	americaines	decouverte
liked	hardly		britannique	informatique	couverture

TABLE 5.1: Example of a set of 11 reasonable points (Word dataset).

5.6.1.4 Reasonable points analysis

Finally, one may wonder what kind of words are selected as reasonable points in the models. The intuition is that they should be some sort of "discriminative prototypes" the classifier is based on. To investigate this, using $K_{\mathbf{C}}$ and a training set of 1,200 examples, we learned a classifier $\boldsymbol{\alpha}$ with a high value of λ to enforce a very sparse model, thus making the analysis easier. The set of 11 reasonable points automatically selected during the learning process is shown in Table 5.1. Our interpretation of why these particular words were chosen is that this small set actually carries a lot of discriminative patterns. Table 5.2 shows some of these patterns (extracted by hand from the reasonable points of Table 5.1) along with their number of occurrences in each class over the entire dataset. For example, words ending with *ly* correspond to English words, while those ending with *que* characterize French words. Note that Table 5.1 also reflects the fact that English words are shorter on average (6.99) than French words (8.26) in the dataset, but the English (resp. French) reasonable points are significantly shorter (resp. longer) than the average (mean of 5.00 and 10.83 resp.), which allows better discrimination. Note that we generated other sets of reasonable points from several training sets and observed the same patterns.

5.6.2 Handwritten Digits

We use the same NIST Special Database 3 as earlier in this manuscript. We have seen that classifying digits using a Freeman code representation and edit similarities

Patterns	`w`	`y`	`k`	`q`	`nn`	`gh`	`ai`	`ed$`	`ly$`	`es?$`	`ques?$`	`^h`
English	146	144	83	14	5	34	39	151	51	265	0	62
French	7	19	5	72	35	0	114	51	0	630	43	14

TABLE 5.2: Some discriminative patterns extracted from the reasonable points of Table 5.1 (`^`: start of word, `$`: end of word, `?`: 0 or 1 occurrence of preceding letter).

yields close-to-perfect accuracy, even in the multi-class setting. In order to make the comparison between the edit similarities (i-iii) easier, we evaluate them on the binary task of discriminating between even and odd digits. This task is harder due to extreme within-class variability: each class is in fact a "meta-class" containing instances of 5 basic classes of digits. Therefore, every example is highly dissimilar to about 80% of the examples of its own class (e.g., 1's are dissimilar to 5's and 0's are dissimilar to 4's, although they belong to the same class).

5.6.2.1 Convergence rate

Once again, we assess the convergence of the cost learning methods (i and iii). We keep aside 2,000 words as a validation set to tune the parameters (using 5-fold cross-validation and selecting the value offering the best classification accuracy) as well as 2,000 words for testing the models. We build bootstrap samples $\mathcal{T}$ from the remaining 6,000 words to learn the edit costs (5 runs for each size $n_{\mathcal{T}}$), as well as 400 words to train the separator $\boldsymbol{\alpha}$.

Figure 5.11 shows the accuracy and sparsity results of each method with respect to $n_{\mathcal{T}}$, averaged over 5 runs. First of all, we notice that the Levenshtein distance d_{lev} performs nicely on this task (95.19% with a model size of 70) and that p_e is never able to match d_{lev}'s accuracy level (94.94% at best with a model size of 78). In our opinion, this poor performance comes from the fact that p_e does not take advantage of negative pairs. In a context of extreme within-class variability, moving closer examples of the same class without making sure that examples of different class are kept far from each others does not yield an appropriate similarity. On the other hand, our method shows the same general behavior on this task as on the previous one. Indeed, convergence is fast despite the richness of the two classes (only 100 examples to match Levenshtein's accuracy and about 1,000 to reach convergence). Moreover, $K_{\mathbf{C}}$ achieves significantly better performance (95.63% at best with a model size of 57) than both d_{lev} (p-value < 0.05 for $n_{\mathcal{T}} \geq 250$ using a Student's t-test) and p_e (p-value < 0.01 for $n_{\mathcal{T}} > 20$).

5.6.2.2 Pairing strategy and influence of α

Figure 5.12 shows the accuracy and sparsity results obtained for $n_{\mathcal{T}} = 2,000$ with respect to α and the pairing strategies. The performance for d_{lev} and p_e is carried over

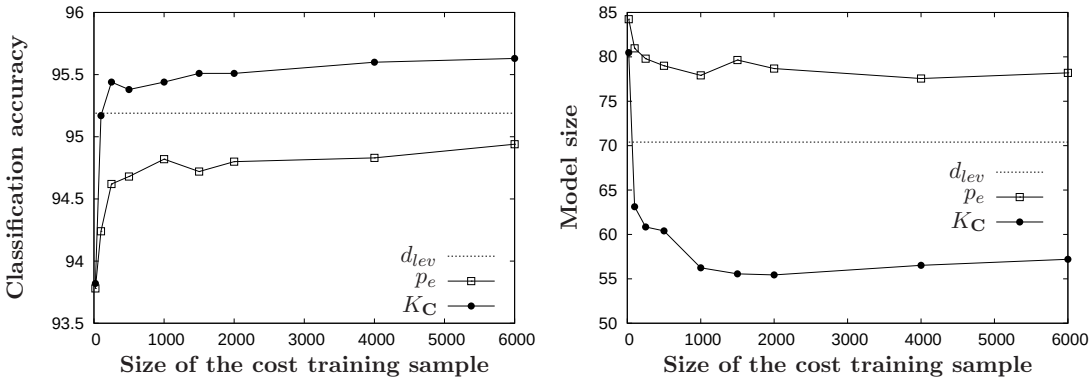


FIGURE 5.11: Learning the edit costs: accuracy and sparsity results (Digit dataset).

from Figure 5.11 for comparison. Results are very different from those obtained on the previous dataset. Here, $K_{\mathbf{C}}$ with random pairing fails: it is always largely outperformed by both d_{lev} and p_e. On the other hand, $K_{\mathbf{C}}$ with Levenshtein pairing performs better than every other approaches for $0.05 \leq \alpha \leq 0.4$. This behavior can be explained by the meta-class structure of the dataset. When using random pairing, many training examples are paired with landmarks of the same class but yet very different (for instance, a 1 paired with a 5, or a 0 paired with a 4), and trying to "move them closer" is a fruitless effort. On the other hand, we have seen earlier that the Levenshtein distance is an appropriate measure to discriminate between handwritten digits. Therefore, when using Levenshtein pairing with $\alpha \leq 0.2$, the problematic situation explained above rarely occurs. When $\alpha > 0.2$, since each meta-class is made out of 5 basic classes in even proportions, more and more examples are paired with "wrong" landmarks and the performance drops dramatically.

This result yields a valuable conclusion: similarity learning should not always focus on optimizing over all possible pairs (although it is often the case in the literature), since it may lead to poor classification performance. In some situations, such as the presence of high within-class variability, it may be a better strategy to improve the similarity according to a few carefully selected pairs.

5.6.2.3 Reasonable points analysis

To provide an insight into the sort of digits that are selected as reasonable points, we follow the same procedure as in Section 5.6.1.4 using a training set of 2,000 examples. We end up with a set of 13 reasonable points. The corresponding digit contours are drawn in Figure 5.13, allowing a graphical interpretation of why these particular examples were chosen. Note that this set is representative of a general tendency: we experimented with several training sets and obtained similar results. The most striking thing about this set is that 7's are over-represented (4 out of the 6 reasonable points of the odd class). This is explained by the fact that 7's (i) account for 1's and 9's (their contour is very similar),

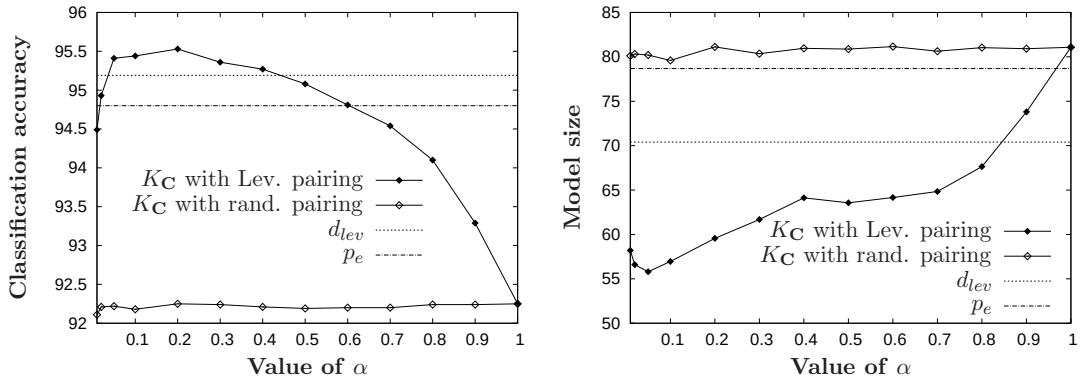


FIGURE 5.12: Pairing strategies: accuracy and sparsity results with respect to α (Digit dataset).

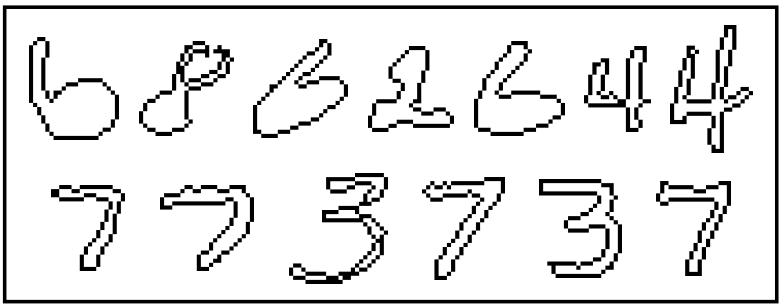


FIGURE 5.13: Example of a set of 13 reasonable points (Digit dataset).

which also gives a reason for the absence of 1's and 9's in the set, and (ii) are not similar to any even digits. The same kind of reasoning applies to 6's (the lower part of 6's is shared by 0's and 8's, but not by any odd number) and 3's (lower part is the same as 5's). We can also notice the presence of 4's: they have a contour mostly made of straight lines, which is unique in the even class. There is also a 2 whose contour is somewhat similar to a 1. Lastly, another explanation for having several occurrences of the same digit may be to account for variations of size (the two 4's), shape or orientations (the three 6's).

5.7 Conclusion

In this chapter, we made use of the theory of (ϵ, γ, τ)-good similarity functions in the context of edit similarities. We first conducted a preliminary experimental study confirming that this framework is well-suited to edit similarities, leading to classification performance competitive with standard SVM but with a number of additional advantages, among which the absence of PSD constraint and the sparsity of the models.

We then went a step further and proposed a novel approach to the problem of learning edit similarities from data, called GESL, driven by the notion of (ϵ, γ, τ)-goodness. As

Method	Data	Model	Scripts	Opt.	Global sol.	Neg. pairs	Gen.
GESL	Strings/Trees	—	Optimal	CO	✓	✓	✓

TABLE 5.3: Summary of the main features of GESL ("Opt.", "Global sol.", "Neg. pairs", "Gen." and "CO" respectively stand for "Optimization", "Global solution", "Negative pairs", "Generalization guarantees" and "Convex optimization").

opposed to most state-of-the-art approaches, GESL is not based on a costly iterative procedure but on solving an efficient convex program, and can accommodate both positive and negative training pairs. Furthermore, it is also a promising way to learn tree edit similarities, even though we did not perform any series of experiments in this case. We provided a theoretical analysis of GESL, which holds for a large class of loss functions. A generalization bound in $O(\sqrt{1/n_{\mathcal{T}}})$ was derived using the notion of uniform stability. This bound is (i) related to the goodness of the resulting similarity, which gives guarantees that the similarity will induce accurate classifiers for the task at hand, and (ii) independent from the size of the alphabet, making GESL suitable for problems involving large vocabularies. We conducted experiments on two string datasets that show that GESL has fast convergence and that the learned similarities perform very well in practice, inducing more accurate and sparser models than other (standard or learned) edit similarities. We also studied two pairing strategies and observed that Levenshtein pairing is more stable to high within-class variability, and that considering all possible pairs is not always a good approach. Table 5.3 summarizes the main features of GESL using the same format as in the survey of Chapter 3 (Table 3.2).

An extension of this work would be to consider sparsity-inducing regularizers on the edit cost matrix. For instance, using an L_1 regularization would lead to more interpretable matrices: an edit cost set to zero during learning would suggest that the corresponding edit operation is not relevant to the task, which can be a valuable information in many real-world applications. This would however prevent the derivation of generalization guarantees using uniform stability, but the theoretical framework presented later in this thesis (Chapter 7) could be used instead.

Another interesting perspective is to assess the relevance of similarities learned with GESL when used in k-Nearest Neighbors classifiers. Indeed, when using Levenshtein pairing, GESL's objective is somewhat related to the k-NN prediction rule and to the objective of the metric learning method LMNN (Weinberger & Saul, 2009). This intuition is confirmed by preliminary results using a 1-NN classifier (see Figure 5.14 and Table 5.4), where $K_{\mathbf{C}}$ outperforms e_L and p_e on both datasets. These first results open the door to a further theoretical analysis and might lead to k-NN generalization guarantees for GESL.

After having dealt with structured data, in the next part of this thesis we will focus on data consisting of feature vectors. While in the context of strings or trees, we could only

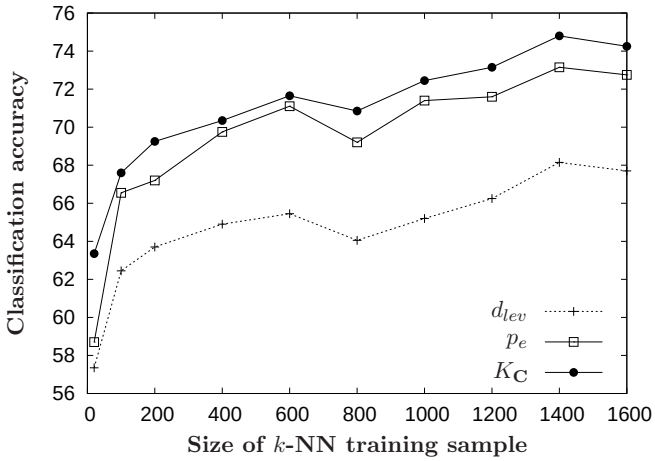


FIGURE 5.14: 1-Nearest Neighbor accuracy results (Word dataset).

Similarity	**Accuracy**
e_L	97.08%
p_e	96.44%
$K_{\mathbf{C}}$	**97.50%**

TABLE 5.4: 1-Nearest Neighbor accuracy results on the Digit dataset.

optimize a pair-based objective that is a loose bound on the empirical (ϵ, γ, τ)-goodness (see Equation 5.3) due to the form of the edit similarity, we will see in the next chapter that using a simple bilinear similarity allows us to optimize the actual (ϵ, γ, τ)-goodness, relying on global constraints instead of pairs.

PART III

Contributions in Metric Learning from Feature Vectors

CHAPTER 6

Learning Good Bilinear Similarities from Global Constraints

Chapter abstract In this chapter, we build upon GESL (proposed in Chapter 5) to learn good similarities between feature vectors. We focus on the bilinear similarity, which is not PSD-constrained. Thanks to this simple form of similarity, we are able to efficiently optimize its empirical (ϵ, γ, τ)-goodness (instead of an upper bound as done in Chapter 5 for structured data) in a nonlinear feature space by formulating the approach as a convex minimization problem. Unlike other metric learning methods, this results in the similarity being optimized with respect to global constraints instead of local pairs or triplets. Then, relying on uniform stability arguments similar to those used in the previous chapter, we derive generalization guarantees directly in terms of the goodness in generalization of the learned similarity. As compared to GESL, our method minimizes a tighter bound on the true risk of the linear classifier built from the similarity. Experiments performed on various datasets confirm the effectiveness of our approach compared to state-of-the-art methods and provide evidence that (i) it is fast, (ii) robust to overfitting and (iii) produces very sparse classifiers.

The material of this chapter is based on the following international publication:

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)*, 2012c.

6.1 Introduction

In the previous chapter, we used a relaxation of the notion of (ϵ, γ, τ)-goodness to propose a pair-based edit similarity learning method and showed that, in this context, we could establish the consistency of the learned metric with respect to unseen pairs of examples, and a relation to the goodness in generalization of the metric. In this chapter, we focus on metric learning from feature vectors and aim at optimizing the *exact* criterion of (ϵ, γ, τ)-goodness. Thanks to the simple form of the bilinear similarity, we are able to do this in an efficient way, leading to a similarity optimized with respect to global constraints (rather than local pairs) and used to build a global linear classifier. Our approach, called SLLC (Similarity Learning for Linear Classification), has several

advantages: (i) it is tailored to linear classifiers, (ii) theoretically well-founded, (iii) does not require positive semi-definiteness, and (iv) is in a sense less restrictive than pair or triplet-based settings. We formulate the problem of learning a good similarity function as a convex minimization problem that can be efficiently solved in a batch or online way. Furthermore, by using the Kernel Principal Component Analysis (KPCA) trick (Chatpatanasiri et al., 2010), we are able to kernelize our algorithm and thereby learn more powerful similarity functions and classifiers in the nonlinear feature space induced by a kernel. From the theoretical standpoint, we show that our approach has uniform stability, which leads to generalization guarantees directly in terms of the (ϵ, γ, τ)-goodness in generalization of the learned similarity. In other words, our approach minimizes an upper bound on the true risk of the linear classifier built from the similarity, and this bound is tighter than that obtained for GESL in Chapter 5. Lastly, we provide an experimental study on seven datasets of various domains and compare SLLC with two widely-used metric learning approaches: LMNN (Weinberger & Saul, 2009) and ITML (Davis et al., 2007). This study demonstrates the practical effectiveness of our method and shows that it is fast, robust to overfitting and induces very sparse classifiers, making it suitable for dealing with high-dimensional data.

The rest of the chapter is organized as follows. Section 6.2 presents our approach, SLLC, and the KPCA trick used to kernelize it. In Section 6.3, we provide a theoretical analysis of SLLC, leading to the derivation of generalization guarantees both in terms of the consistency of the learned similarity and the error of the linear classifier. Finally, Section 6.4 features an experimental study on various datasets and we conclude in Section 6.5.

6.2 Learning (ϵ, γ, τ)-Good Bilinear Similarity Functions

We consider the bilinear similarity $K_{\mathbf{M}}$ defined by

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

In order to satisfy $K_{\mathbf{M}} \in [-1, 1]$, we assume that inputs are normalized such that $||\mathbf{x}||_2 \leq 1$, and we require $||\mathbf{M}||_{\mathcal{F}} \leq 1$.

6.2.1 Similarity Learning Formulation

Our goal is to directly optimize the empirical (ϵ, γ, τ)-goodness of $K_{\mathbf{M}}$. To this end, we are given a training sample of $n_{\mathcal{T}}$ labeled points $\mathcal{T} = \{z_i = (\mathbf{x_i}, y_i)\}_{i=1}^{n_{\mathcal{T}}}$ and a sample of $n_{\mathcal{R}}$ labeled reasonable points $\mathcal{R} = \{z_k = (\mathbf{x_k}, y_k)\}_{k=1}^{n_{\mathcal{R}}}$. In practice, $\mathcal{R}$ is a subset of $\mathcal{T}$ with $n_{\mathcal{R}} = \hat{\tau} n_{\mathcal{T}}$ ($\hat{\tau} \in \ ]0, 1]$). In the lack of background knowledge, it can be drawn randomly or according to some criterion, e.g., diversity (Kar & Jain, 2011).

Based on the definition of (ϵ, γ, τ)-goodness in hinge loss (Definition 5.3), given $\mathcal{R}$ and a margin γ, we want to optimize the amount of margin violation ϵ on the training sample (the empirical goodness). Thus, let

$$\ell(\mathbf{M}, z_i, \mathcal{R}) = [1 - y_i \frac{1}{\gamma n_{\mathcal{R}}} \sum_{k=1}^{n_{\mathcal{R}}} y_k K_{\mathbf{M}}(\mathbf{x_i}, \mathbf{x_k})]_+$$

denote the empirical goodness of $K_{\mathbf{M}}$ with respect to a single training point z_i. The empirical goodness over the sample $\mathcal{T}$ is denoted by

$$\epsilon_{\mathcal{T}} = \frac{1}{n_{\mathcal{T}}} \sum_{i=1}^{n_{\mathcal{T}}} \ell(\mathbf{M}, z_i, \mathcal{R}).$$

We want to learn the matrix $\mathbf{M}$ that minimizes $\epsilon_{\mathcal{T}}$. This can be done by solving the following regularized problem, referred to as SLLC (Similarity Learning for Linear Classification):

$$\min_{\mathbf{M} \in \mathbb{R}^{d \times d}} \quad \epsilon_{\mathcal{T}} \quad + \quad \beta \|\mathbf{M}\|_{\mathcal{F}}^2 \tag{6.1}$$

where β is a regularization parameter.

Note that SLLC can be cast as a convex quadratic program (QP) by rewriting the sum of $n_{\mathcal{T}}$ hinge losses in the objective function as $n_{\mathcal{T}}$ margin constraints and introducing slack variables $\boldsymbol{\xi} \in \mathbb{R}_+^{n_{\mathcal{T}}}$ in the objective:

$$\begin{aligned} \min_{\mathbf{M} \in \mathbb{R}^{d \times d}, \boldsymbol{\xi} \in \mathbb{R}_+^{n_{\mathcal{T}}}} \quad & \frac{1}{n_{\mathcal{T}}} \sum_{i=1}^{n_{\mathcal{T}}} \xi_i \quad + \quad \beta \|\mathbf{M}\|_{\mathcal{F}}^2 \\ \text{s.t.} \quad & 1 - y_i \frac{1}{\gamma n_{\mathcal{R}}} \sum_{k=1}^{n_{\mathcal{R}}} y_k K_{\mathbf{M}}(\mathbf{x_i}, \mathbf{x_k}) \leq \xi_i, \quad 1 \leq i \leq n_{\mathcal{T}}. \end{aligned} \tag{6.2}$$

SLLC is radically different from classic metric and similarity learning algorithms presented in Chapter 3, which are based on pair or triplet-based constraints. It learns a global similarity rather than a local one, since $\mathcal{R}$ is the same for each training example. Moreover, the constraints are easier to satisfy since they are defined over an average of similarity scores to the points in $\mathcal{R}$ instead of over a single pair or triplet. This means that one can fulfill a constraint without satisfying the margin for each point in $\mathcal{R}$ individually (unlike what we did with GESL in Chapter 5). SLLC also has a number of desirable properties:

1. No costly semi-definite programming is required, as opposed to many Mahalanobis distance learning methods. In its convex QP form (6.2), SLLC can be solved efficiently using standard convex minimization solvers. Moreover, it has only one constraint per training example (instead of one for each pair or triplet), i.e., a total

of only $n_{\mathcal{T}}$ constraints and $n_{\mathcal{T}} + d^2$ variables. In its unconstrained form (6.1), it is convex but not differentiable everywhere due to the hinge function in the loss. It can be solved in a stochastic or online setting using composite objective mirror descent (Duchi et al., 2010) or dual averaging methods (Xiao, 2010) and thereby scales to very large problems.

2. The size of $\mathcal{R}$ does not affect the complexity of Problem 6.2, since each constraint is simply a linear combination of entries of $\mathbf{M}$.

3. If $\mathbf{x_i}$ is sparse, then the associated constraint is sparse as well: some variables of the problem (corresponding to entries of $\mathbf{M}$) have a zero coefficient in the constraint. This makes the problem easier to solve when data have a sparse representation.

We now explain how SLLC can be kernelized to deal with nonlinear problems.

6.2.2 Kernelization of SLLC

The framework presented in the previous section is theoretically well-founded with respect to Balcan et al.'s theory and has some generalization guarantees, as we will see in the next section. Moreover, it has the advantage of being very simple: we learn a global linear similarity and use it to build a global linear classifier. In order to learn more powerful similarities (and therefore classifiers), we propose to kernelize the approach by learning them in the nonlinear feature space induced by a kernel.

As discussed in Section 3.2.3, kernelizing a particular metric learning algorithm is difficult in general and may lead to intractable problems unless dimensionality reduction is applied. For these reasons, we instead use the KPCA trick, recently proposed by Chatpatanasiri et al. (2010). It provides a straightforward way to kernelize a metric learning algorithm while performing dimensionality reduction at no additional cost, and is based on Kernel Principal Component Analysis (Schölkopf et al., 1998), a nonlinear extension of PCA (Pearson, 1901).

PCA provides a way of representing the data by a small number k of linearly uncorrelated variables (called the principal components) that account for most of the variance in the data. Assuming zero-centered data, let $\mathbf{C}$ denote the data covariance matrix:

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x_i} \mathbf{x_j}^T.$$

The new representation $\mathbf{x'_i} \in \mathbb{R}^k$ $(k \leq d)$ of a data point $\mathbf{x_i} \in \mathbb{R}^d$ is given by $\mathbf{x'_i} = \mathbf{x_i}^T \mathbf{V}$, where $\mathbf{V}$ is a matrix whose columns are the top k eigenvectors of $\mathbf{C}$.

The basic idea of KPCA is to use a kernel function to implicitly perform PCA in the (possibly infinite-dimensional) nonlinear feature space induced by the kernel, in the spirit

of what is done in SVM. Let K be a kernel such that $K(x, y) = \langle \phi(x), \phi(x') \rangle$. The data covariance matrix in the new feature space is given by

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i)\phi(x_j)^T.$$

It can be shown that the projection of a point $\phi(x_i)$ onto the jth principal component only depends on inner products and therefore can be computed implicitly through the kernel function. The solution can actually be obtained through an eigendecomposition of the kernel matrix $\mathbf{K}$ whose entries are defined as $K_{i,j} = K(x_i, x_j)$.

Therefore, KPCA allows us to project the data into a new feature space of dimension $k \leq n$. The (unchanged) metric learning algorithm can then be used to learn a metric in that nonlinear space. Chatpatanasiri et al. (2010) showed that the KPCA trick is theoretically sound for unconstrained metric learning algorithms (they proved representer theorems), which includes SLLC. Throughout the rest of this chapter, we will only consider the kernelized version of SLLC.

Generally speaking, kernelizing a metric learning algorithm may cause or increase overfitting, especially when data are scarce and/or high-dimensional. However, since our framework is entirely linear and global, we expect our method to be quite robust to this undesirable effect. This will be doubly confirmed in the rest of this chapter: experimentally in Section 6.4, but also theoretically with the derivation in the following section of generalization guarantees independent from the size of the projection space.

6.3 Theoretical Analysis

In this section, we present a theoretical analysis of our approach. Our main result is the derivation of a generalization bound (Theorem 6.4) guaranteeing the consistency of SLLC and thus the (ϵ, γ, τ)-goodness in generalization for the considered task.

6.3.1 Notations

For convenience, given a bilinear model $K_{\mathbf{M}}$, we denote by $\mathbf{M}_{\mathcal{R}}$ both the similarity defined by the matrix $\mathbf{M}$ and its associated set of reasonable points $\mathcal{R}$ (when it is clear from the context we may omit the subscript $\mathcal{R}$). Given a similarity $\mathbf{M}_{\mathcal{R}}$, $\ell(\mathbf{M}_{\mathcal{R}}, z, \mathcal{R})$ is the loss function over one example z. The empirical risk of $\mathbf{M}_{\mathcal{R}}$ over the sample $\mathcal{T}$ is thus given by

$$R^{\ell}_{\mathcal{T}}(\mathbf{M}_{\mathcal{R}}) = \epsilon_{\mathcal{T}}(\mathbf{M}_{\mathcal{R}}) = \frac{1}{n_{\mathcal{T}}} \sum_{i=1}^{n_{\mathcal{T}}} \ell(\mathbf{M}_{\mathcal{R}}, z_i, \mathcal{R})$$

and corresponds to the empirical goodness, while the true risk is given by

$$R^{\ell}(\mathbf{M}_{\mathcal{R}}) = \epsilon(\mathbf{M}_{\mathcal{R}}) = \mathbb{E}_{z\sim P}[\ell(\mathbf{M}_{\mathcal{R}}, z, \mathcal{R})]$$

and corresponds to the "true" goodness (or goodness in generalization). In the following, we will rather use $\epsilon_{\mathcal{T}}(\mathbf{M}_{\mathcal{R}})$ and $\epsilon(\mathbf{M}_{\mathcal{R}})$ to denote respectively the empirical and true risks to highlight the equivalence between risk and (ϵ, γ, τ)-goodness in SLLC. When it is clear from the context, we may simply use $\epsilon_{\mathcal{T}}$ and ϵ.

The similarity is optimized according to a fixed set $\mathcal{R}$ of reasonable points coming from the training sample. Therefore, these reasonable points may not follow the distribution from which the training sample has been generated. Once again, the framework of uniform stability allows us to cope with this situation. Note that the empirical and true risks are defined with respect to a single example and not with respect to pairs. Therefore, we use the standard uniform stability setting (presented in Section 2.2.2) instead of the adaptation to the pair-based case introduced by Jin et al. (2009) and used in Chapter 5.

6.3.2 Generalization Bound

In our case, to prove the uniform stability property we need to show that

$$\forall \mathcal{T}, \forall i, \sup_z |\ell(\mathbf{M}, z, \mathcal{R}) - \ell(\mathbf{M}^i, z, \mathcal{R}^i)| \leq \frac{\kappa}{n_{\mathcal{T}}}, \tag{6.3}$$

where $\mathbf{M}$ is learned from $\mathcal{T}$ and $\mathcal{R} \subseteq \mathcal{T}$, $\mathbf{M}^i$ is the matrix learned from $\mathcal{T}^i$ and $\mathcal{R}^i \subseteq \mathcal{T}^i$ is the set of reasonable points associated to $\mathcal{T}^i$. $\mathcal{T}^i$ is obtained from $\mathcal{T}$ by replacing the ith example $z_i \in \mathcal{T}$ by another example z_i' independent from $\mathcal{T}$ and drawn from P. Note that $\mathcal{R}$ and $\mathcal{R}^i$ are of equal size and can differ in at most one example, depending on whether z_i or z_i' belong to their corresponding set of reasonable points. For the sake of simplicity, we assume that ℓ is bounded by 1.[1] To show (6.3), we need the following results.

Lemma 6.1. *For any labeled examples $z = (\mathbf{x}, y)$, $z' = (\mathbf{x}', y')$ and any models $\mathbf{M}_{\mathcal{R}}$, $\mathbf{M}'_{\mathcal{R}'}$, the following properties hold:*

P1: $|K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')| \leq 1$,

P2: $|K_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') - K_{\mathbf{M}'}(\mathbf{x}, \mathbf{x}')| \leq \|\mathbf{M} - \mathbf{M}'\|_{\mathcal{F}}$,

P3: 1-admissibility property of ℓ:

$$|\ell(\mathbf{M}, z, \mathcal{R}) - \ell(\mathbf{M}', z, \mathcal{R}')| \leq 1 \left|\frac{\sum_{k=1}^{n_{\mathcal{R}}} y_k K_{\mathbf{M}}(\mathbf{x}, \mathbf{x_k})}{\gamma n_{\mathcal{R}}} - \frac{\sum_{j=1}^{n_{\mathcal{R}'}} y'_k K_{\mathbf{M}'}(\mathbf{x}, \mathbf{x'_k})}{\gamma n_{\mathcal{R}'}}\right|.$$

[1] Since we assume $\|\mathbf{x}\|_2 \leq 1$ and $\|\mathbf{M}\|_{\mathcal{F}} \leq 1$, this can be obtained by dividing ℓ by the constant $1 + \frac{1}{\gamma}$.

Proof. $P1$ comes from $|K_{\mathbf{M}}(\mathbf{x},\mathbf{x}')| \leq \|\mathbf{x}\|_2\|\mathbf{M}\|_{\mathcal{F}}\|\mathbf{x}'\|_2$, the normalization on examples ($\|\mathbf{x}\|_2 \leq 1$) and the requirement on matrices ($\|\mathbf{M}\|_{\mathcal{F}} \leq 1$).

For $P2$, we observe that

$$|K_{\mathbf{M}}(\mathbf{x},\mathbf{x}') - K_{\mathbf{M}'}(\mathbf{x},\mathbf{x}')| = |K_{\mathbf{M}-\mathbf{M}'}(\mathbf{x},\mathbf{x}')|,$$

and we use the normalization $\|\mathbf{x}\|_2 \leq 1$.

$P3$ follows directly from $|y| = 1$ and the 1-lipschitz property of the hinge loss:

$$|[U]_+ - [V]_+| \leq |U - V|.$$

□

Let $F_{\mathcal{T}} = \epsilon_{\mathcal{T}}(\mathbf{M}) + \beta\|\mathbf{M}\|_{\mathcal{F}}^2$ be the objective function of SLLC with respect to a sample $\mathcal{T}$ and a set of reasonable points $R \subseteq T$. The following lemma bounds the deviation between $\mathbf{M}$ and $\mathbf{M}^i$.

Lemma 6.2. *For any models* $\mathbf{M}$ *and* $\mathbf{M^i}$ *that are minimizers of* $F_{\mathcal{T}}$ *and* $F_{\mathcal{T}^i}$ *respectively, we have:*

$$\|\mathbf{M} - \mathbf{M^i}\|_{\mathcal{F}} \leq \frac{1}{\beta n_{\mathcal{T}}\gamma}.$$

Proof. We follow closely the proof of Lemma 20 of Bousquet & Elisseeff (2002) and omit some details for the sake of readability (similar ideas are used in the first part of the more detailed proof of Lemma 5.8). Let $\Delta\mathbf{M} = \mathbf{M^i} - \mathbf{M}$, $0 \leq t \leq 1$ and

$$\begin{aligned} M_1 &= \|\mathbf{M}\|_{\mathcal{F}}^2 - \|\mathbf{M} + t\Delta\mathbf{M}\|_{\mathcal{F}}^2 + \|\mathbf{M}^i\|_{\mathcal{F}}^2 - \|\mathbf{M}^i - t\Delta\mathbf{M}\|_{\mathcal{F}}^2 \\ M_2 &= \frac{1}{\beta n_{\mathcal{T}}}(\epsilon_{\mathcal{T}}(\mathbf{M}_{\mathcal{R}}) - \epsilon_{\mathcal{T}}((\mathbf{M} + t\Delta\mathbf{M})_{\mathcal{R}}) + \epsilon_{\mathcal{T}^i}((\mathbf{M} + t\Delta\mathbf{M})_{\mathcal{R}}) - \epsilon_{\mathcal{T}^i}(\mathbf{M}_{\mathcal{R}})). \end{aligned}$$

Using the fact that $F_{\mathcal{T}}$ and $F_{\mathcal{T}^i}$ are convex functions, that $\mathbf{M}$ and $\mathbf{M}^i$ are their respective minimizers and property P3, we have $M_1 \leq M_2$. Fixing $t = 1/2$, we obtain $M_1 = \|\mathbf{M} - \mathbf{M}^i\|_{\mathcal{F}}^2$, and using property $P3$ and the normalization $\|\mathbf{x}\|_2 \leq 1$, we get:

$$M_2 \leq \frac{1}{\beta n_{\mathcal{T}}\gamma}(\|\frac{1}{2}\Delta\mathbf{M}\|_{\mathcal{F}} + \|-\frac{1}{2}\Delta\mathbf{M}\|_{\mathcal{F}}) = \frac{\|\mathbf{M} - \mathbf{M}^i\|_{\mathcal{F}}}{\beta n_{\mathcal{T}}\gamma}.$$

This leads to the inequality $\|\mathbf{M} - \mathbf{M}^i\|_{\mathcal{F}}^2 \leq \frac{\|\mathbf{M}-\mathbf{M}^i\|_{\mathcal{F}}}{\beta n_{\mathcal{T}}\gamma}$ from which Lemma 6.2 is directly derived. □

We now have all the material needed to prove the stability property of our algorithm.

Lemma 6.3. *Let* $n_{\mathcal{T}}$ *and* $n_{\mathcal{R}}$ *be the number of training examples and reasonable points respectively,* $n_{\mathcal{R}} = \hat{\tau} n_{\mathcal{T}}$ *with* $\hat{\tau} \in]0,1]$. *SLLC has a uniform stability in* $\frac{\kappa}{n_{\mathcal{T}}}$ *with* $\kappa = \frac{1}{\gamma}(\frac{1}{\beta\gamma} + \frac{2}{\hat{\tau}}) = \frac{\hat{\tau}+2\beta\gamma}{\hat{\tau}\beta\gamma^2}$, *where* β *is the regularization parameter and* γ *the margin.*

Proof. For any sample $\mathcal{T}$ of size $n_{\mathcal{T}}$, any $1 \leq i \leq n_{\mathcal{T}}$, any labeled examples $z = (\mathbf{x}, y)$ and $z'_i = (\mathbf{x'_i}, y'_i) \sim P$:

$$
\begin{aligned}
&|\ell(\mathbf{M}, z, \mathcal{R}) - \ell(\mathbf{M}^i, z, \mathcal{R}^i)| \\
&\leq \left| \frac{1}{\gamma n_{\mathcal{R}}} \sum_{k=1}^{n_{\mathcal{R}}} y_k K_{\mathbf{M}}(\mathbf{x}, \mathbf{x_k}) - \frac{1}{\gamma n_{\mathcal{R}^i}} \sum_{k=1}^{n_{\mathcal{R}^i}} y_k K_{\mathbf{M}^i}(\mathbf{x}, \mathbf{x_k}) \right| \\
&= \left| \frac{1}{\gamma n_{\mathcal{R}}} \left(\left(\sum_{k=1, k\neq i}^{n_{\mathcal{R}}} y_k (K_{\mathbf{M}}(\mathbf{x}, \mathbf{x_k}) - K_{\mathbf{M}^i}(\mathbf{x}, \mathbf{x_k})) \right) + \right. \right. \\
&\qquad \left. \left. y_i K_{\mathbf{M}}(\mathbf{x}, \mathbf{x_i}) - y'_i K_{\mathbf{M}^i}(\mathbf{x}, \mathbf{x_i}') \right) \right| \\
&\leq \frac{1}{\gamma n_{\mathcal{R}}} \left(\left(\sum_{k=1, k\neq i}^{n_{\mathcal{R}}} (|y_k| \|\mathbf{M} - \mathbf{M}^i\|_{\mathcal{F}}) \right) + \right. \\
&\qquad \left. |y_i K_{\mathbf{M}^i}(\mathbf{x}, \mathbf{x_i})| + |y'_i K_{\mathbf{M}}(\mathbf{x}, \mathbf{x'_i})| \right) \\
&\leq \frac{1}{\gamma n_{\mathcal{R}}} \left(\frac{n_{\mathcal{R}} - 1}{\beta n_{\mathcal{T}} \gamma} + 2 \right) \leq \frac{1}{\gamma n_{\mathcal{R}}} \left(\frac{n_{\mathcal{R}}}{\beta n_{\mathcal{T}} \gamma} + 2 \right).
\end{aligned}
$$

The first inequality follows from $P3$. The second comes from the fact that $\mathcal{R}$ and $\mathcal{R}^i$ differ in at most one element, corresponding to the example z_i in $\mathcal{R}$ and the example z'_i replacing z_i in $\mathcal{R}^i$. The last inequalities are obtained by the use of the triangle inequality, $P1$, $P2$, Lemma 6.2, and the fact that the labels belong to $\{-1, 1\}$. Since $n_{\mathcal{R}} = \hat{\tau} n_{\mathcal{T}}$, we get $|\ell(\mathbf{M}, z, \mathcal{R}) - \ell(\mathbf{M}^i, z, \mathcal{R}^i)| \leq \frac{1}{\gamma n_{\mathcal{T}}}(\frac{1}{\beta\gamma} + \frac{2}{\hat{\tau}})$. □

Applying Theorem 5.9 with Lemma 6.2 gives our main result.

Theorem 6.4. *Let $\gamma > 0$, $\delta > 0$ and $n_{\mathcal{T}} > 1$. With probability at least $1 - \delta$, for any model $\mathbf{M}_R$ learned with SLLC, we have:*

$$
\epsilon \leq \epsilon_{\mathcal{T}} + \frac{1}{n_{\mathcal{T}}} \left(\frac{\hat{\tau} + 2\beta\gamma}{\hat{\tau}\beta\gamma^2} \right) + \left(\frac{2(\hat{\tau} + 2\beta\gamma)}{\hat{\tau}\beta\gamma^2} + 1 \right) \sqrt{\frac{\ln 1/\delta}{2n_{\mathcal{T}}}}.
$$

Theorem 6.4 highlights three important properties of SLLC. First, it has a reasonable $O(1/\sqrt{n_{\mathcal{T}}})$ convergence rate. Second, it is independent from the dimensionality of the data. This is due to the fact that $\|\mathbf{M}\|_{\mathcal{F}}$ is bounded by a constant. Third, Theorem 6.4 bounds the true goodness of the learned similarity function. By minimizing $\epsilon_{\mathcal{T}}$ with SLLC, we minimize ϵ and thus an upper bound on the true risk of the resulting linear classifier, as stated by Theorem 5.4. Note that this is a much tighter bound on the

goodness than that derived in Chapter 5, where only a loose bound on the empirical goodness was optimized.

6.4 Experimental Validation

We propose a comparative study of our method against two widely-used Mahalanobis distance learning algorithms: Large Margin Nearest Neighbor[2] (LMNN) from Weinberger & Saul (2009) and Information-Theoretic Metric Learning[3] (ITML) from Davis et al. (2007). Recall that LMNN essentially optimizes the k-NN error on the training set (with a safety margin), whereas ITML aims at best satisfying pair-based constraints while minimizing the LogDet divergence between the learned matrix $\mathbf{M}$ and the identity matrix (refer to Section 3.2.1 for more details on these methods). We conduct this experimental study on seven classic binary classification datasets of varying domain, size and difficulty, mostly taken from the UCI Machine Learning Repository[4]. Their properties are summarized in Table 6.1. Some of them, such as Breast, Ionosphere or Pima, have already been extensively used to evaluate metric learning methods.

6.4.1 Setup

We compare the following methods: (i) the cosine similarity $K_{\mathbf{I}}$ in KPCA space, as a baseline, (ii) SLLC, (iii) LMNN in the original space, (iv) LMNN in KPCA space, (v) ITML in the original space, and (vi) ITML in KPCA space.[5] All attributes are scaled to $[-1/d; 1/d]$ to ensure $\|\mathbf{x}\|_2 \leq 1$.

To generate a new feature space using KPCA, we use the Gaussian kernel with parameter σ equal to the mean of all pairwise training data Euclidean distances (a standard heuristic, used for instance by Kar & Jain, 2011). Ideally, we would like to project the data to the feature space of maximum size (equal to the number of training examples), but to keep the computations tractable we only retain three times the number of features of the original data (four times for the low-dimensional datasets), as shown in Table 6.1.[6] On Cod-RNA, KPCA was run on a randomly drawn subsample of 10% of the training data.

Unless predefined training and test sets are available (as for Splice, Svmguide1 and Cod-RNA), we randomly generate 70/30 splits of the data, and average the results over 100 runs. Training sets are further partitioned 70/30 for validation purposes.

[2]Code download from: http://www.cse.wustl.edu/~kilian/code/lmnn/lmnn.html
[3]Code download from: http://www.cs.utexas.edu/~pjain/itml/
[4]http://archive.ics.uci.edu/ml/
[5]$K_{\mathbf{I}}$, LMNN and ITML are normalized to ensure their values belong to $[-1, 1]$.
[6]Note that the amount of variance captured thereby was greater than 90% for all datasets.

Dataset	Breast	Iono.	Rings	Pima	Splice	Svmguide1	Cod-RNA
# training examples	488	245	700	537	1,000	3,089	59,535
# test examples	211	106	300	231	2,175	4,000	271,617
# dimensions	9	34	2	8	60	4	8
# dim. after KPCA	27	102	8	24	180	16	24
# runs	100	100	100	100	1	1	1

TABLE 6.1: Properties of the seven datasets used in the experimental study.

We tune the following parameters by cross-validation: $\beta, \gamma \in \{10^{-7}, \ldots, 10^{-2}\}$ for SLLC, $\lambda_{ITML} \in \{10^{-4}, \ldots, 10^{4}\}$ for ITML, and $\lambda \in \{10^{-3}, \ldots, 10^{2}\}$ for learning the linear classifiers, choosing the value offering the best accuracy. We choose $\mathcal{R}$ to be the entire training set, i.e., $\hat{\tau} = 1$ (interestingly, cross-validation of $\hat{\tau}$ did not improve the results significantly). We take $k = 3$ and $\mu = 0.5$ for LMNN, as suggested by Weinberger & Saul (2009). For ITML, we generate $n_{\mathcal{T}}$ random constraints for a fair comparison with SLLC.

6.4.2 Results

Linear classification We first report the results obtained in linear classification using Balcan's learning rule (Table 6.2). SLLC achieves the highest accuracy on 5 out of 7 datasets and competitive performance on the remaining 2. At the same time, on all datasets, SLLC leads to extremely sparse classifiers. The sparsity of the classifier corresponds to the number of training examples that are involved in classifying a new example. Therefore, SLLC leads to much simpler and yet often more accurate classifiers than those built from other similarities. Furthermore, sparsity allows faster predictions, especially when data are plentiful and/or high-dimensional (e.g., Cod-RNA or Splice). Often enough, the learned linear classifier has sparsity 1, which means that classifying a new example boils down to computing its similarity score to a single training example and compare the value with a threshold. Note that we tried large values of λ to obtain sparser classifiers from $K_{\mathbf{I}}$, LMNN and ITML, but this yielded dramatic drops in accuracy. The extreme sparsity brought by SLLC comes from the fact that the constraints are based on an average of similarity scores over the same set of points for all training examples. This brings to the fore the relevance of optimizing the similarity with respect to global constraints.

Nearest neighbor classification Since LMNN and ITML are designed for k-NN use, we also give the results obtained in 3-NN classification (Table 6.3). Surprinsingly (because it is not designed for k-NN), SLLC achieves the best results on 4 datasets (a possible reason for this is given in the next paragraph). It is, however, outperformed by LMNN or ITML on the 3 biggest problems. For most tasks, the accuracy obtained in linear classification is better or similar to that of 3-NN (highlighting the fact that

Dataset	Breast	Iono.	Rings	Pima	Splice	Svmguide1	Cod-RNA
$K_{\mathbf{I}}$	96.57 *20.39*	89.81 *52.93*	100.00 *18.20*	75.62 *25.93*	83.86 *362*	**96.95** ***64***	**95.91** ***557***
SLLC	**96.90** ***1.00***	**93.25** ***1.00***	**100.00** ***1.00***	**75.94** ***1.00***	**87.36** ***1***	96.55 *8*	94.08 *1*
LMNN	96.81 *9.98*	90.21 *13.30*	100.00 *18.04*	75.15 *69.71*	85.61 *315*	95.80 *157*	88.40 *61*
LMNN KPCA	96.01 *8.46*	86.12 *9.96*	100.00 *8.73*	74.92 *22.20*	86.85 *156*	96.53 *82*	95.15 *591*
ITML	96.80 *9.79*	92.09 *9.51*	100.00 *17.85*	75.25 *56.22*	81.47 *377*	96.70 *49*	95.06 *164*
ITML KPCA	96.23 *17.17*	93.05 *18.01*	100.00 *15.21*	75.25 *16.40*	85.29 *287*	96.55 *89*	95.14 *206*

TABLE 6.2: Average accuracy (normal type) and sparsity (italic type) of the linear classifiers built from the studied similarity functions. For each dataset, boldface indicates the most accurate method (sparsity is used to break the ties).

Dataset	Breast	Iono.	Rings	Pima	Splice	Svmguide1	Cod-RNA
$K_{\mathbf{I}}$	96.71	83.57	**100.00**	72.78	77.52	93.93	90.07
SLLC	**96.90**	**93.25**	**100.00**	**75.94**	87.36	93.82	94.08
LMNN	96.46	88.68	**100.00**	72.84	83.49	96.23	94.98
LMNN KPCA	96.23	87.13	**100.00**	73.50	**87.59**	95.85	94.43
ITML	92.67	88.29	**100.00**	72.07	77.43	95.97	**95.42**
ITML KPCA	96.38	87.56	**100.00**	72.80	84.41	**96.80**	95.32

TABLE 6.3: Average accuracy of 3-NN classifiers using the studied similarity functions. For each dataset, boldface indicates the most accurate method.

metric learning for linear classification is of interest) while prediction is many orders of magnitude faster due to the sparsity of the linear separators. Also note that an accurate similarity for k-NN classification can achieve poor results in linear classification (LMNN on Cod-RNA), and vice versa (SLLC on Svmguide1).

Robustness to overfitting SLLC's good performance on small datasets can be credited to its robustness to overfitting. Indeed, LMNN and ITML are optimized with respect to local constraints, which tend to get easier to satisfy simultaneously as dimensionality grows. On the other hand, SLLC is optimized with respect to global constraints and can thus be seen as more robust. This is confirmed by Figure 6.1, which shows the accuracy of SLLC, LMNN and ITML on the Ionosphere dataset with respect to the number of dimensions retained in KPCA. As expected, LMNN and ITML, tend to overfit as the dimensionality grows while SLLC suffers from very limited overfitting.

Visualization of the projection space Recall that in Balcan's learning rule, the similarity is used to build a similarity map: data are projected into a new feature

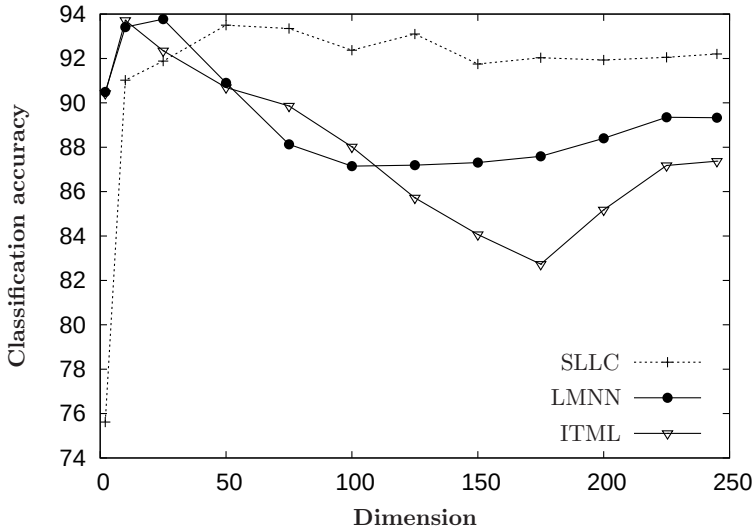


FIGURE 6.1: Accuracy of the methods with respect to the dimensionality of the KPCA space on Ionosphere.

space where each coordinate corresponds to the similarity score to a training example, and a linear classifier is learned in that space. Figure 6.2 and Figure 6.3 show a low-dimensional embedding of the feature space induced by each similarity for the Rings and Svmguide1 datasets respectively. On both datasets, the space induced by SLLC is the most appropriate to linear classification: the data is well-separated even in this 2D representation of the space. On the Rings dataset, the data is actually perfectly separated in 1D, which explains why we achieve perfect classification accuracy relying on 1 training instance only. This highlights the fact that SLLC optimizes a criterion which is designed for linear classification, and its potential for dimensionality reduction. Conversely, the feature spaces induced by $K_{\mathbf{I}}$, LMNN and ITML do not offer such quality of linear separability — for instance and unsurprisingly, LMNN tends to induce spaces that are better suited to nearest neighbor classification.

Runtime comparison In this series of experiments, SLLC was solved in its QP form using the standard convex minimization solver MOSEK[7] while LMNN and ITML have their own specific and sophisticated solver. Despite this fact, SLLC is several orders of magnitude faster than LMNN (see Table 6.4) because its number of constraints is much smaller. However, it remains slower than ITML.

[7] http://www.mosek.com/

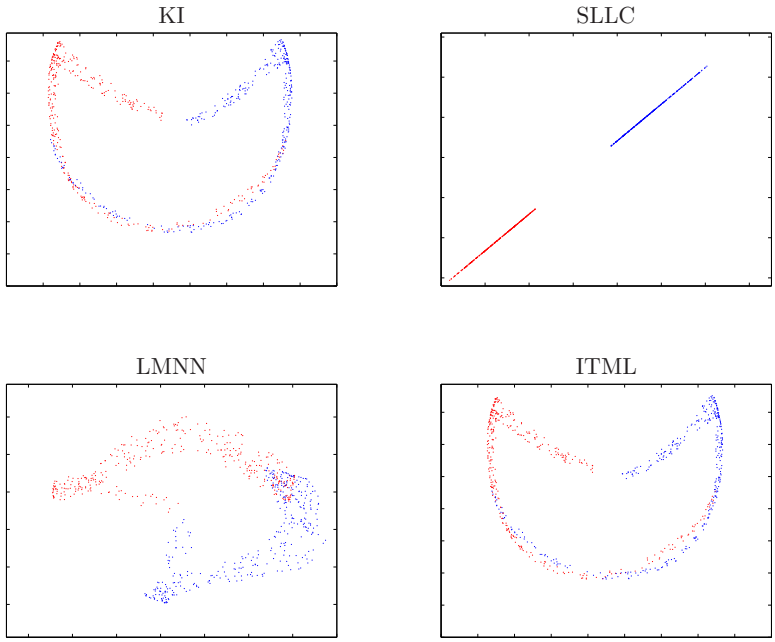


FIGURE 6.2: Feature space induced by the similarity in which the linear classifier is learned (Rings dataset). Dimension was reduced to 2 for visualization purposes using Principal Component Analysis.

Dataset	Breast	Iono.	Rings	Pima	Splice	Svmguide1	Cod-RNA
SLLC	4.76	5.36	0.05	4.01	158.38	185.53	2471.25
LMNN	25.99	16.27	37.95	32.14	309.36	331.28	10418.73
LMNN KPCA	41.06	34.57	84.86	48.28	1122.60	369.31	24296.41
ITML	2.09	3.09	0.19	2.96	3.41	0.83	5.98
ITML KPCA	1.68	5.77	0.20	2.74	56.14	5.30	25.25

TABLE 6.4: Average time per run (in seconds) required for learning the similarity.

6.5 Conclusion

In this chapter, we presented SLLC, a novel approach to bilinear similarity learning that makes use of both the theory of (ϵ, γ, τ)-goodness and the KPCA trick. It is formulated as a convex minimization problem that can be solved efficiently using standard techniques. We derived a generalization bound based on the notion of uniform stability that is independent from the size of the input space, and thus from the number of dimensions selected by KPCA. It guarantees the true goodness of the learned similarity, and therefore our method can be seen as minimizing an upper bound on the true risk of the linear classifier built from the learned similarity. We experimentally demonstrated the effectiveness of SLLC and also showed that the learned similarities induce extremely sparse classifiers. Combined with the independence from dimensionality and the robustness to overfitting, it makes the approach very efficient and suitable for high-dimensional data.

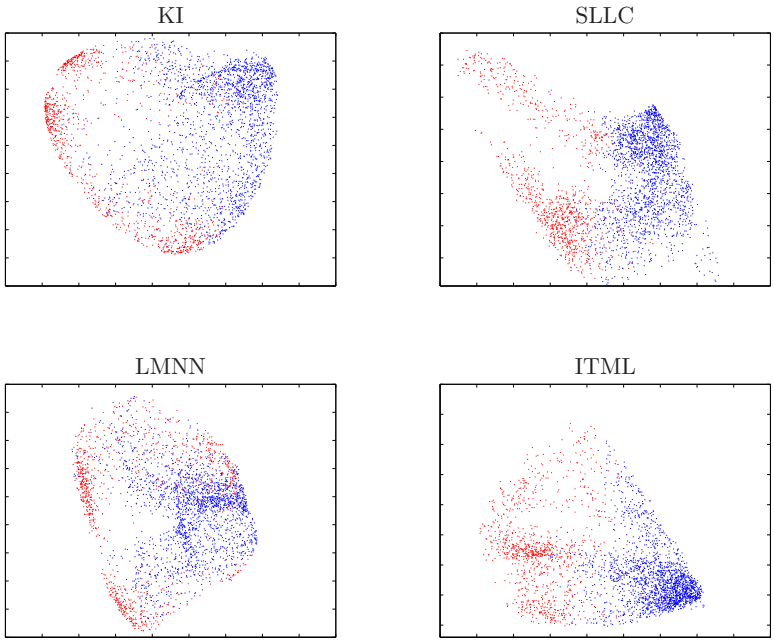


FIGURE 6.3: Feature space induced by the similarity in which the linear classifier is learned (Svmguide1 dataset). Dimension was reduced to 2 for visualization purposes using Principal Component Analysis.

Method	Convex	Scalable	Competitive	Reg.	Low-rank	Online	Gen.
SLLC	✓	✓✓✓	✓	✓	✗	✓	✓

TABLE 6.5: Summary of the main features of SLLC ("Reg." and "Gen." respectively stand for "Regularized" and "Generalization guarantees").

Table 6.5 summarizes the main features of SLLC using the same format as in the survey of Chapter 3 (Table 3.1).

It would be interesting to investigate the performance of SLLC when solved in its unconstrained form, either in a stochastic or online way. This would dramatically improve its runtime on large-scale problems and hopefully not significantly reduce the classification performance.

As shown in Table 6.5, SLLC is not a low-rank approach, since Frobenius norm regularization does not favor low-rank matrices. Another promising perspective would be to study the influence of other regularizers on $\mathbf{A}$, in particular the trace norm or the $L_{2,1}$ norm that tend to induce such matrices. Recent advances in stochastic and online optimization of problems regularized with these norms (Duchi et al., 2010; Xiao, 2010; Yang et al., 2010) could be used to derive an efficient algorithm. The use of such norms would add sparsity at the metric level in addition to the sparsity already obtained at the classifier level.

However, recall that the generalization of such formulations cannot be studied using stability-based arguments, since sparse algorithms are known not to be stable. On the other hand, algorithmic robustness can deal with such algorithms more easily. In the next chapter, we propose an adaptation of robustness to the metric learning setting.

CHAPTER 7
Robustness and Generalization for Metric Learning

Chapter abstract Throughout this thesis, we have argued that little work has been done about the generalization ability of metric learning algorithms. We made use in Chapter 5 and Chapter 6 of uniform stability arguments to derive generalization guarantees for our metric learning methods. Unfortunately, these arguments are somewhat limited to the use of Frobenius regularizarion and thus cannot be applied to many existing metric learning algorithms, in particular those using a sparse or low-rank regularizer on the metric. In this chapter, we address this theoretical issue by proposing an adaptation of the notion of algorithmic robustness (previously introduced by Xu and Mannor) to the classic metric learning setting, where training data consist of pairs or triplets. We show that if a metric learning algorithm is robust in our sense, then it has generalization guarantees. We further show that a weak notion of robustness is a necessary and sufficient condition for an algorithm to generalize, justifying that it is fundamental to metric learning. Lastly, we illustrate how our framework can be used to derive generalization bounds for a large class of metric learning algorithms, some of which could not be studied using previous approaches.

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

Aurélien Bellet and Amaury Habrard. Robustness and Generalization for Metric Learning. Technical report, University of Saint-Etienne, September 2012. arXiv:1209.1086.

7.1 Introduction

Most of the research effort in metric learning has gone into formulating the problem as tractable optimization procedures, but very little has been done on the generalization ability of learned metrics on unseen data, due to the fact that the training pairs/triplets are not i.i.d. As we have seen in Section 3.2.1.4, online metric learning methods (e.g., Shalev-Shwartz et al., 2004; Jain et al., 2008; Chechik et al., 2009) offer some guarantees, but only in the form of regret bounds assuming that the algorithm is provided with i.i.d. pairs/triplets, and say nothing about generalization to unseen data. Conversion of regret bounds into batch generalization bounds is possible (see for instance Cesa-Bianchi et al., 2001, 2004) but as a consequence these bounds also require the i.i.d. assumption.

Putting aside our contributions in Chapter 5 and Chapter 6, the question of the generalization ability of batch metric learning has only been addressed in two recent papers, described in Section 3.2.1.5. For the sake of readability, we recall here their main features. The approach of Bian & Tao (2011; 2012) uses a statistical analysis to give generalization guarantees for loss minimization methods, but their results rely on some hypotheses on the distribution of the examples and do not take into account any regularization on the metric. The most general contribution was proposed by Jin et al. (2009) who adapted the framework of uniform stability to regularized metric learning. However, their approach is based on Frobenius norm regularization and cannot be applied to many types of regularization, in particular sparsity-inducing norms (Xu et al., 2012a).

In this last contribution, we propose to address the lack of theoretical framework by studying the generalization ability of metric learning algorithms according to a notion of algorithmic robustness. Recall that algorithmic robustness, introduced by Xu & Mannor (2010, 2012) and described in Section 2.2.3, allows one to derive generalization bounds when, given two "close" training and testing examples, the variation between their associated loss is bounded. This notion of closeness of examples relies on a partition of the input space into different regions such that two examples in the same region are seen as close. We propose here to adapt this notion of algorithmic robustness to metric learning, where training data is made of pairs (or triplets). We show that, in the context of robustness, the problem of training pairs not being i.i.d. can be worked around by simply assuming that the pairs are built from an i.i.d. sample of labeled examples. Moreover, following the work of Xu & Mannor (2010, 2012), we establish that a weaker notion robustness is actually necessary and sufficient for metric learning algorithms to generalize, highlighting that robustness is a fundamental property. Lastly, we illustrate the applicability of our framework by deriving generalization bounds for a larger class of problems than Jin et al. (2009), using very few algorithm-specific arguments. In particular, it can accommodate a vast choice of regularizers and unlike the approach of Bian & Tao (2011; 2012), requires no assumption on the distribution of the examples.

The rest of the chapter is organized as follows. Our notion of algorithmic robustness for metric learning is presented in Section 7.2. The necessity and sufficiency of weak robustness is shown in Section 7.3. Section 7.4 is devoted to the application of the proposed framework: we show that a large class of metric learning algorithms are robust. Finally, we conclude in Section 7.5.

7.2 Robustness and Generalization for Metric Learning

After introducing some notations and assumptions, we present our definition of robustness for metric learning and show that if a metric learning algorithm is robust, then it

has generalization guarantees.

7.2.1 Preliminaries

We assume that the instance space $\mathcal{X}$ is a compact convex metric space with respect to a norm $\|\cdot\|$ such that $\mathcal{X} \subset \mathbb{R}^d$, thus there exists a constant R such that $\forall \mathbf{x} \in \mathcal{X}$, $\|\mathbf{x}\| \leq R$. A metric is a function $f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Recall that we use the generic term metric to refer to a distance or a (dis)similarity function.

Given a training sample $\mathcal{T} = \{z_i = (\mathbf{x_i}, y_i)\}_{i=1}^n$ drawn i.i.d. from an unknown joint distribution P over the space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, we denote by $\mathcal{P}_\mathcal{T}$ the set of all possible pairs built from $\mathcal{T}$:

$$\mathcal{P}_\mathcal{T} = \{(z_1, z_1), \cdots, (z_1, z_n), \cdots, (z_n, z_n)\}.$$

We generally assume that a metric learning algorithm $\mathcal{A}$ takes as input a finite set of pairs from $(\mathcal{Z} \times \mathcal{Z})^n$ and outputs a metric. We denote by $\mathcal{A}_\mathcal{P}$ the metric learned by an algorithm $\mathcal{A}$ from a sample $\mathcal{P}$ of pairs. With any pair of labeled examples (z, z') and any metric f, we associate a loss function $\ell(f, z, z')$ that depends on the examples and their labels. This loss is assumed to be nonnegative and uniformly bounded by a constant B. We define the true risk of f by

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

We denote the empirical risk of f over the sample of pairs $\mathcal{P}$ by

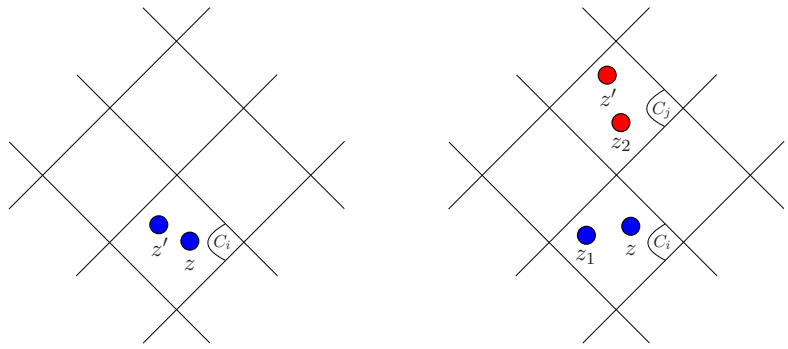
$$R^\ell_\mathcal{P}(f) = \frac{1}{|\mathcal{P}|} \sum_{(z_i, z_j) \in \mathcal{P}} \ell(f, z_i, z_j).$$

On a few occasions, we discuss the extension of our framework to triplet-based metric learning, where an algorithm $\mathcal{A}$ takes as input a finite set of triplets from $(\mathcal{Z} \times \mathcal{Z} \times \mathcal{Z})^n$. Instead of considering all pairs $\mathcal{P}_\mathcal{T}$ built from $\mathcal{T}$, we consider the sample of *admissible* triplets $\mathcal{R}_\mathcal{T}$ built from $\mathcal{T}$ such that for any $(z_1, z_2, z_3) \in \mathcal{R}_\mathcal{T}$, z_1 and z_2 share the same label while z_3 does not, with the interpretation that z_1 must be more similar to z_2 than to z_3. In this context, the loss function ℓ is defined with respect to triplets of examples and the true risk of a metric f is given by

$$R^\ell(f) = \mathbb{E}_{\substack{z,z',z'' \sim P \\ y = y' \neq y''}}[\ell(f, z, z', z'')]$$

and the empirical risk of f over the sample of admissible triplets $\mathcal{R}$ by

$$R^\ell_\mathcal{R}(f) = \frac{1}{|\mathcal{R}|} \sum_{(z_i, z_j, z_k) \in \mathcal{R}} \ell(f, z_i, z_j, z_k).$$



Classic robustness Robustness for metric learning

FIGURE 7.1: Illustration of the property of robustness in the classic and metric learning settings. In this example, we use a cover based on the L_1 norm. In the classic definition, if any example z' falls in the same region C_i as a training example z, then the deviation between their loss must be bounded. In the metric learning definition proposed in this work, for any pair (z, z') and a training pair (z_1, z_2), if z, z_1 belong to some region C_i and z', z_2 to some region C_j, then the deviation between the loss of these two pairs must be bounded.

7.2.2 Robustness for Metric Learning

We present here our adaptation of the definition of robustness to metric learning.

In Section 2.2.3, we have seen that robustness relies on a partition of the space $\mathcal{Z}$ into K disjoint subsets such that for every training and testing instances belonging to the same region of the partition, the deviation between their respective losses is bounded by a term $\epsilon(\mathcal{T})$.[1] In order to adapt this notion to metric learning, the idea is to use the partition of $\mathcal{Z}$ at the pair level: if a new test pair of examples is close to a training pair, then the respective losses of the two pairs must be close. Two pairs are close when each instance of the first pair falls into the same subset of the partition of $\mathcal{Z}$ as the corresponding instance of the other pair, as shown in Figure 7.1. A metric learning algorithm with this property is called robust. This notion is formalized in the following definition.

Definition 7.1 (Robustness for metric learning)**.** An algorithm $\mathcal{A}$ is $(K, \epsilon(\cdot))$ robust for $K \in \mathbb{N}$ and $\epsilon(\cdot) : (\mathcal{Z} \times \mathcal{Z})^n \to \mathbb{R}$ if $\mathcal{Z}$ can be partitioned into K disjoints sets, denoted by $\{C_i\}_{i=1}^K$, such that the following holds for all $\mathcal{T} \in \mathcal{Z}^n$:
$\forall (z_1, z_2) \in \mathcal{P}_\mathcal{T}, \forall z, z' \in \mathcal{Z}, \forall i, j \in [K]$: if $z_1, z \in C_i$ and $z_2, z' \in C_j$ then

$$|\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_1, z_2) - \ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')| \leq \epsilon(\mathcal{P}_\mathcal{T}).$$

[1] Recall from Section 2.2.3 that $\mathcal{Z}$ is partitioned such that if two examples fall into the same region, then they share the same label.

K and $\epsilon(\cdot)$ quantify the robustness of the algorithm which depends on the training sample. Note that the property of robustness is required for every training pair of the sample — we will later see that this property can be relaxed.

Note that this definition of robustness can be easily extended to triplet-based metric learning. In this context, the robustness property can then be expressed by:
$\forall (z_1, z_2, z_3) \in \mathcal{R}_{\mathcal{T}}, \forall z, z', z'' \in \mathcal{Z}, \forall i, j \in [K]$: if $z_1, z \in C_i$, $z_2, z' \in C_j$, $z_3, z'' \in C_k$ then

$$|\ell(\mathcal{A}_{\mathcal{R}_{\mathcal{T}}}, z_1, z_2, z_3) - \ell(\mathcal{A}_{\mathcal{R}_{\mathcal{T}}}, z, z', z'')| \leq \epsilon(\mathcal{R}_{\mathcal{T}}). \tag{7.1}$$

7.2.3 Generalization of Robust Metric Learning Algorithms

We now give a PAC generalization bound for metric learning algorithms satisfying the property of robustness (Definition 7.1). We first give the following concentration inequality that we will use in the derivation of the bound.

Proposition 7.2 (van der Vaart & Wellner, 2000)**.** *Let* $(|N_1|, \ldots, |N_K|)$ *an i.i.d. multinomial random variable with parameters* n *and* $(\mu(C_1), \ldots, \mu(C_K))$*. By the Breteganolle-Huber-Carol inequality we have:* $Pr\left\{\sum_{i=1}^{K} \left|\frac{|N_i|}{n} - \mu(C_i)\right| \geq \lambda\right\} \leq 2^K \exp\left(\frac{-n\lambda^2}{2}\right)$*, hence with probability at least* $1-\delta$*,*

$$\sum_{i=1}^{K} \left|\frac{N_i}{n} - \mu(C_i)\right| \leq \sqrt{\frac{2K \ln 2 + 2\ln(1/\delta)}{n}}. \tag{7.2}$$

We now give our first result on the generalization of metric learning algorithms.

Theorem 7.3. *If a learning algorithm* $\mathcal{A}$ *is* $(K, \epsilon(\cdot))$*-robust and the training sample consists of the pairs* $\mathcal{P}_{\mathcal{T}}$ *obtained from a sample* $\mathcal{T}$ *generated by* n *i.i.d. draws from* P*, then for any* $\delta > 0$*, with probability at least* $1-\delta$ *we have:*

$$|R^{\ell}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}}})| \leq \epsilon(\mathcal{P}_{\mathcal{T}}) + 2B\sqrt{\frac{2K \ln 2 + 2\ln(1/\delta)}{n}}.$$

Proof. Let N_i be the set of index of points of $\mathcal{T}$ that fall into the C_i. $(|N_1|, \ldots, |N_K|)$ is an i.i.d. random variable with parameters n and $(\mu(C_1), \ldots, \mu(C_K))$. We have:

$$
\begin{aligned}
&|R^\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}) - R^\ell_{\mathcal{P}_\mathcal{T}}(\mathcal{A}_{\mathcal{P}_\mathcal{T}})| \\
&= \left|\sum_{i=1}^K\sum_{j=1}^K \mathbb{E}_{z,z'\sim P}(\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')|z \in C_i, z' \in C_j)\mu(C_i)\mu(C_j) - \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_i, z_j)\right| \\
&\overset{(a)}{\leq} \left|\sum_{i=1}^K\sum_{j=1}^K \mathbb{E}_{z,z'\sim P}(\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')|z \in C_i, z' \in C_j)\mu(C_i)\mu(C_j) - \right. \\
&\qquad\qquad \left.\sum_{i=1}^K\sum_{j=1}^K \mathbb{E}_{z,z'\sim P}(\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')|z \in C_i, z' \in C_j)\mu(C_i)\frac{|N_j|}{n}\right| + \\
&\quad \left|\sum_{i=1}^K\sum_{j=1}^K \mathbb{E}_{z,z'\sim P}(\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')|z \in C_i, z' \in C_j)\mu(C_i)\frac{|N_j|}{n} - \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_i, z_j)\right| \\
&\overset{(b)}{\leq} \left|\sum_{i=1}^K\sum_{j=1}^K \mathbb{E}_{z,z'\sim P}(\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')|z \in C_i, z' \in C_j)\mu(C_i)(\mu(C_j) - \frac{|N_j|}{n})\right| + \\
&\quad \left|\sum_{i=1}^K\sum_{j=1}^K \mathbb{E}_{z,z'\sim P}(\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')|z \in C_i, z' \in C_j)\mu(C_i)\frac{|N_j|}{n} - \right. \\
&\qquad\qquad \left.\sum_{i=1}^K\sum_{j=1}^K \mathbb{E}_{z,z'\sim P}(\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')|z \in C_i, z' \in C_j)\frac{|N_i|}{n}\frac{|N_j|}{n}\right| + \\
&\quad \left|\sum_{i=1}^K\sum_{j=1}^K \mathbb{E}_{z,z'\sim P}(\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')|z \in C_i, z' \in C_j)\frac{|N_i|}{n}\frac{|N_j|}{n} - \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_i, z_j)\right| \\
&\overset{(c)}{\leq} B\left(\left|\sum_{j=1}^K \mu(C_j) - \frac{|N_j|}{n}\right| + \left|\sum_{i=1}^K \mu(C_i) - \frac{|N_i|}{n}\right|\right) + \\
&\quad \left|\frac{1}{n^2}\sum_{i=1}^K\sum_{j=1}^K \sum_{z_o\in N_i}\sum_{z_l\in N_j} \max_{z\in C_i}\max_{z'\in C_j} |\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z') - \ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_o, z_l)|\right| \\
&\overset{(d)}{\leq} \epsilon(\mathcal{P}_\mathcal{T}) + 2B\sum_{i=1}^K\left|\frac{|N_i|}{n} - \mu(C_i)\right| \\
&\overset{(e)}{\leq} \epsilon(\mathcal{P}_\mathcal{T}) + 2B\sqrt{\frac{2K\ln 2 + 2\ln(1/\delta)}{n}}.
\end{aligned}
$$

Inequalities (a) and (b) are due to the triangle inequality, (c) uses the fact that ℓ is bounded by B, that $\sum_{i=1}^K \mu(C_i) = 1$ by definition of a multinomial random variable and that $\sum_{j=1}^K \frac{|N_j|}{n} = 1$ by definition of the N_j. Lastly, (d) comes from the definition of robustness (Definition 7.1) and (e) from the application of Proposition 7.2. □

The previous bound depends on K which is given by the cover chosen for $\mathcal{Z}$. If for any K, the associated $\epsilon(\cdot)$ is constant with respect to $\mathcal{T}$ (i.e., $\epsilon_K(\mathcal{T}) = \epsilon_K$), we can prove a

bound holding uniformly for all K:

$$|R^\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}) - R^\ell_{\mathcal{P}_\mathcal{T}}(\mathcal{A}_{\mathcal{P}_\mathcal{T}})| \leq \inf_{K\geq 1}\left[\epsilon_K + 2B\sqrt{\frac{2K\ln 2 + 2\ln(1/\delta)}{n}}\right].$$

The bound also gives an insight into what should be the objective of a robust metric learning algorithm: according to a partition of the labeled input space, given two regions, minimize the maximum loss over pairs of examples belonging to each region.

For triplet-based metric learning algorithms, by following the definition of robustness given by (7.1) and adapting straightforwardly the losses to triplets such that they output zero for non-admissible triplets, Theorem 7.3 can be easily extended to obtain the following generalization bound:

$$|R^\ell(\mathcal{A}_{\mathcal{R}_\mathcal{T}}) - R^\ell_{\mathcal{R}_\mathcal{T}}(\mathcal{A}_{\mathcal{R}_\mathcal{T}})| \leq \epsilon(\mathcal{R}_\mathcal{T}) + 3B\sqrt{\frac{2K\ln 2 + 2\ln(1/\delta)}{n}}. \tag{7.3}$$

7.2.4 Pseudo-robustness

The previous study requires the robustness property to be satisfied for every training pair. We show, with the following definition, that it is possible to relax the robustness to be fulfilled for only a subpart of the training sample and yet be able to derive generalization guarantees.

Definition 7.4. An algorithm $\mathcal{A}$ is $(K, \epsilon(\cdot), \hat{p}_n(\cdot))$ pseudo-robust for $K \in \mathbb{N}$, $\epsilon(\cdot) : (\mathcal{Z}\times\mathcal{Z})^n \to \mathbb{R}$ and $\hat{p}_n(\cdot) : (\mathcal{Z}\times\mathcal{Z})^n \to \{1,\ldots,n^2\}$, if $\mathcal{Z}$ can be partitioned into K disjoints sets, denoted by $\{C_i\}_{i=1}^K$, such that for all $\mathcal{T} \in \mathcal{Z}^n$ i.i.d. from P, there exists a subset of training pairs samples $\hat{\mathcal{P}}_\mathcal{T} \subseteq \mathcal{P}_\mathcal{T}$, with $|\hat{\mathcal{P}}_\mathcal{T}| = \hat{p}_n(\mathcal{P}_\mathcal{T})$, such that the following holds:
$\forall (z_1, z_2) \in \hat{\mathcal{P}}_\mathcal{T}, \forall z, z' \in \mathcal{Z}, \forall i, j \in [K]$: if $z_1, z \in C_i$ and $z_2, z' \in C_j$ then

$$|\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_1, z_2) - l(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')| \leq \epsilon(\mathcal{P}_\mathcal{T}). \tag{7.4}$$

We can easily observe that $(K, \epsilon(\cdot))$-robust is equivalent to $(K, \epsilon(\cdot), n^2)$ pseudo-robust. The following theorem illustrates the generalization guarantees associated to the pseudo-robustness property.

Theorem 7.5. *If a learning algorithm $\mathcal{A}$ is $(K, \epsilon(\cdot), \hat{p}_n(\cdot))$ pseudo-robust and the training pairs $\mathcal{P}_\mathcal{T}$ come from a sample generated by n i.i.d. draws from P, then for any $\delta > 0$, with probability at least $1-\delta$ we have:*

$$|R^\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}) - R^\ell_{\mathcal{P}_\mathcal{T}}(\mathcal{A}_{\mathcal{P}_\mathcal{T}})| \leq \frac{\hat{p}_n(\mathcal{P}_\mathcal{T})}{n^2}\epsilon(\mathcal{P}_\mathcal{T}) + B\left(\frac{n^2 - \hat{p}_n(\mathcal{P}_\mathcal{T})}{n^2} + 2\sqrt{\frac{2K\ln 2 + 2\ln(1/\delta)}{n}}\right).$$

Proof. The proof is similar to that of Theorem 7.3 and is given in Appendix B.2.1. □

The notion of pseudo-robustness characterizes a situation that often occurs in metric learning: it is difficult to satisfy pair-based constraints for all possible pairs. Theorem 7.5 shows that it is sufficient to satisfy a property of robustness over only a subset of the pairs to have generalization guarantees. Moreover, it also gives an insight into the behavior of metric learning approaches aiming at learning a distance to be plugged in a k-NN classifier such as LMNN (Weinberger & Saul, 2009). These methods do not optimize the distance according to all possible pairs, but only according to the nearest neighbors of the same class and some pairs of different class. According to the previous theorem, this strategy is well-founded provided that the robustness property is fulfilled for some of the pairs used to optimize the metric. Finally, note that this notion of pseudo-robustness can be also easily adapted to triplet based metric learning.

7.3 Necessity of Robustness

We prove here that a notion of weak robustness is actually necessary and sufficient to generalize in a metric learning setup. This result is based on an asymptotic analysis following the work of Xu & Mannor (2012). We consider pairs of instances coming from an increasing sample of training instances $\mathcal{T} = (z_1, z_2, \dots)$ and from a sample of test instances $\mathcal{U} = (z'_1, z'_2, \dots)$ such that both samples are assumed to be drawn i.i.d. from some distribution P. We use $\mathcal{T}(n)$ and $\mathcal{U}(n)$ to denote the first n examples of $\mathcal{T}$ and $\mathcal{U}$ respectively, while $\mathcal{T}^*$ denotes a fixed sequence of training examples.

We first define a notion of generalizability for metric learning.

Definition 7.6 (Generalizability for metric learning)**.** Given a training pair set $\mathcal{P}_{\mathcal{T}^*}$ built from a sequence of examples $\mathcal{T}^*$, a metric learning method $\mathcal{A}$ generalizes with respect to $\mathcal{P}_{\mathcal{T}^*}$ if

$$\lim_n \left| R^\ell(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^\ell_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) \right| = 0.$$

A learning method $\mathcal{A}$ generalizes with probability 1 if it generalizes with respect to the pairs $\mathcal{P}_{\mathcal{T}}$ of almost all samples $\mathcal{T}$ i.i.d. from P.

Note that this notion of generalizability implies convergence in mean. We then introduce the notion of weak robustness for metric learning.

Definition 7.7 (Weak robustness for metric learning)**.** Given a set of training pairs $\mathcal{P}_{\mathcal{T}^*}$ built from a sequence of examples $\mathcal{T}^*$, a metric learning method $\mathcal{A}$ is weakly robust with respect to $\mathcal{P}_{\mathcal{T}^*}$ if there exists a sequence of $\{\mathcal{D}_n \subseteq \mathcal{Z}^n\}$ such that $Pr(\mathcal{U}(n) \in \mathcal{D}_n) \to 1$ and

$$\lim_n \left\{ \max_{\hat{\mathcal{T}}(n) \in \mathcal{D}_n} \left| R^\ell_{\mathcal{P}_{\hat{\mathcal{T}}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^\ell_{\mathcal{P}_{\mathcal{T}^*(n)}} \mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) \right| \right\} = 0.$$

A learning method $\mathcal{A}$ is almost surely weakly robust if it is robust with respect to almost all $\mathcal{T}$.

The definition of robustness requires the labeled sample space to be partitioned into disjoint subsets such that if some instances of pairs of train/test examples belong to the same partition, then they have similar loss. Weak robustness is a generalization of this notion where we consider the average loss of testing and training pairs: if for a large (in the probabilistic sense) subset of data, the testing loss is close to the training loss, then the algorithm is weakly robust. From Proposition 7.2, we can see that if for any fixed $\epsilon > 0$ there exists K such that an algorithm $\mathcal{A}$ is $(K, \epsilon(\cdot))$ robust, then $\mathcal{A}$ is weakly robust. We now give the main result of this section about the necessity of robustness.

Theorem 7.8. *Given a fixed sequence of training examples $\mathcal{T}^*$, a metric learning method $\mathcal{A}$ generalizes with respect to $\mathcal{P}_{\mathcal{T}^*}$ if and only if it is weakly robust with respect to $\mathcal{P}_{\mathcal{T}^*}$.*

Proof Following Xu & Mannor (2012), the sufficiency is obtained by the fact that the testing pairs are built from a sample $\mathcal{U}(n)$ made of n i.i.d. instances. We give the proof in Appendix B.2.2.

For the necessity, we need the following lemma which is a direct adaptation of Lemma 2 from Xu & Mannor (2012). We provide the proof in Appendix B.2.3 for the sake of completeness.

Lemma 7.9. *Given $\mathcal{T}^*$, if a learning method is not weakly robust with respect to $\mathcal{P}_{\mathcal{T}^*}$, there exist $\epsilon^*, \delta^* > 0$ such that the following holds for infinitely many n:*

$$Pr(|R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})| \geq \epsilon^*) \geq \delta^*. \tag{7.5}$$

Now, recall that ℓ is nonnegative and uniformly bounded by B, thus by the McDiarmid inequality (Theorem 5.10) we have that for any $\epsilon, \delta > 0$ there exists an index n^* such that for any $n > n^*$, with probability at least $1 - \delta$, we have:

$$\left| \frac{1}{n^2} \sum_{(z'_i, z'_j) \in \mathcal{P}_{\mathcal{U}(n)}} \ell(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}, z'_i, z'_j) - R^{\ell}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) \right| \leq \epsilon.$$

This implies the convergence

$$R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) \xrightarrow{Pr} 0,$$

and thus from a given index:

$$|R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})| \leq \frac{\epsilon^*}{2}. \tag{7.6}$$

Now, by contradiction, suppose algorithm $\mathcal{A}$ is not weakly robust, Lemma 7.9 implies Equation 7.5 holds for infinitely many n. This combined with Equation 7.6 implies that for infinitely many n:

$$|R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})| \geq \frac{\epsilon^*}{2}$$

which means $\mathcal{A}$ does not generalize, thus the necessity of weak robustness is established. □

The following corollary follows immediately from Theorem 7.8.

Corollary 7.10. *A metric learning method $\mathcal{A}$ generalizes with probability 1 if and only if it is almost surely weakly robust.*

This corollary establishes a strong link between generalization in metric learning and the notion of weak robustness. In the next section, we illustrate the applicability of our framework by showing that many existing metric learning algorithms are robust in our sense.

7.4 Examples of Robust Metric Learning Algorithms

We first restrict our attention to Mahalanobis distance learning algorithms of the form:

$$\min_{\mathbf{M}\succeq 0} \quad \frac{1}{n^2} \sum_{(z_i,z_j)\in\mathcal{P}_{\mathcal{T}}} \ell(d^2_{\mathbf{M}}, z_i, z_j) \quad + \quad C\|\mathbf{M}\|, \tag{7.7}$$

where $\|\cdot\|$ is some matrix norm and $C > 0$ a regularization parameter. The loss function ℓ is assumed to be of the form

$$\ell(d^2_{\mathbf{M}}, z_i, z_j) = g(y_i y_j[1 - d^2_{\mathbf{M}}(\mathbf{x_i}, \mathbf{x_j})]),$$

where g is nonnegative and Lipschitz continuous with Lipschitz constant U. It typically outputs a small value when its input is large positive and a large value when it is large negative. Lastly, $g_0 = \sup_{z,z'} g(y_i y_j[1 - d^2_{\mathbf{M}}(\mathbf{0}, \mathbf{x}, \mathbf{x}')])$ is the largest loss when $\mathbf{M}$ is the zero matrix $\mathbf{0}$.

Recall that showing that a metric learning algorithm is robust (Definition 7.1) implies that the algorithm has generalization guarantees (Theorem 7.3). To prove the robustness of (7.7), we will use the following theorem, which essentially says that if a metric learning algorithm achieves approximately the same testing loss for pairs that are close to each other, then it is robust.

Theorem 7.11. *Fix $\gamma > 0$ and a metric ρ of $\mathcal{Z}$. Suppose that $\forall z_1, z_2, z, z' : (z_1, z_2) \in \mathcal{P}_\mathcal{T}, \rho(z_1, z) \leq \gamma, \rho(z_2, z') \leq \gamma$, $\mathcal{A}$ satisfies*

$$|\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_1, z_2) - \ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')| \leq \epsilon(\mathcal{P}_\mathcal{T}),$$

and $\mathcal{N}(\gamma/2, \mathcal{Z}, \rho) < \infty$. Then $\mathcal{A}$ is $(\mathcal{N}(\gamma/2, \mathcal{Z}, \rho), \epsilon(\mathcal{P}_\mathcal{T}))$-robust.

Proof. By definition of covering number, we can partition $\mathcal{X}$ in $\mathcal{N}(\gamma/2, \mathcal{X}, \rho)$ subsets such that each subset has a diameter less or equal to γ. Furthermore, since $\mathcal{Y}$ is a finite set, we can partition $\mathcal{Z}$ into $|\mathcal{Y}|\mathcal{N}(\gamma/2, \mathcal{X}, \rho)$ subsets $\{C_i\}$ such that $z_1, z \in C_i \Rightarrow \rho(z_1, z) \leq \gamma$. Therefore,

$$|\ell(\mathcal{A}_{p_\mathcal{T}}, z_1, z_2) - \ell(\mathcal{A}_{p_\mathcal{T}}, z, z')| \leq \epsilon(\mathcal{P}_\mathcal{T}), \quad \forall z_1, z_2, z, z' : (z_1, z_2) \in \mathcal{P}_\mathcal{T}, \rho(z_1, z) \leq \gamma, \rho(z_2, z') \leq \gamma$$

implies $z_1, z_2 \in \mathcal{P}_\mathcal{T}, z_1, z \in C_i, z_2, z' \in C_j \Rightarrow |\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_1, z_2) - \ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')| \leq \epsilon(\mathcal{P}_\mathcal{T})$, which establishes the theorem. □

We now prove the robustness of (7.7) when $\|\mathbf{M}\|$ is the Frobenius norm, which corresponds to the formulation (3.6) addressed by Jin et al. (2009).

Example 7.1 (Frobenius norm)**.** *Algorithm* (7.7) *with* $\|\mathbf{M}\| = \|\mathbf{M}\|_\mathcal{F}$ *is* $(|\mathcal{Y}|\mathcal{N}(\gamma/2, \mathcal{X}, \|\cdot\|_2), \frac{8UR\gamma g_0}{C})$*-robust.*

Proof. Let $\mathbf{M}^*$ be the solution given training data $\mathcal{P}_\mathcal{T}$. Due to optimality of $\mathbf{M}^*$, we have

$$\frac{1}{n^2} \sum_{(z_i, z_j) \in \mathcal{P}_\mathcal{T}} g(y_i y_j [1 - d^2_{\mathbf{M}^*}(\mathbf{x_i}, \mathbf{x_j})]) + C\|\mathbf{M}^*\|_\mathcal{F} \leq \frac{1}{n^2} \sum_{(z_i, z_j) \in \mathcal{P}_\mathcal{T}} g(y_i y_j [1 - d^2_{\mathbf{0}}(\mathbf{x_i}, \mathbf{x_j})]) + C\|\mathbf{0}\|_\mathcal{F} = g_0$$

and thus $\|\mathbf{M}^*\|_\mathcal{F} \leq g_0/C$.

We can partition $\mathcal{Z}$ as $|\mathcal{Y}|\mathcal{N}(\gamma/2, \mathcal{X}, \|\cdot\|_2)$ sets, such that if z and z' belong to the same set, then $y = y'$ and $\|\mathbf{x} - \mathbf{x}'\|_2 \leq \gamma$. Now, for $z_1, z_2, z'_1, z'_2 \in \mathcal{Z}$, if $y_1 = y'_1$, $\|\mathbf{x_1} - \mathbf{x'_1}\|_2 \leq \gamma$,

$y_2 = y_2'$ and $\|\mathbf{x_2} - \mathbf{x_2'}\|_2 \leq \gamma$, then:

$$
\begin{aligned}
&|g(y_1 y_2[1 - d^2_{\mathbf{M}^*}(\mathbf{x_1}, \mathbf{x_2})]) - g(y_1' y_2'[1 - d^2_{\mathbf{M}^*}(\mathbf{x_1'}, \mathbf{x_2'})])| \\
&\quad \leq \; U|(\mathbf{x_1} - \mathbf{x_2})^T \mathbf{M}^*(\mathbf{x_1} - \mathbf{x_2}) - (\mathbf{x_1'} - \mathbf{x_2'})^T \mathbf{M}^*(\mathbf{x_1'} - \mathbf{x_2'})| \\
&\quad = \; U|(\mathbf{x_1} - \mathbf{x_2})^T \mathbf{M}^*(\mathbf{x_1} - \mathbf{x_2}) - (\mathbf{x_1} - \mathbf{x_2})^T \mathbf{M}^*(\mathbf{x_1'} - \mathbf{x_2'}) \\
&\qquad + (\mathbf{x_1} - \mathbf{x_2})^T \mathbf{M}^*(\mathbf{x_1'} - \mathbf{x_2'})| - (\mathbf{x_1'} - \mathbf{x_2'})^T \mathbf{M}^*(\mathbf{x_1'} - \mathbf{x_2'})| \\
&\quad = \; U|(\mathbf{x_1} - \mathbf{x_2})^T \mathbf{M}^*(\mathbf{x_1} - \mathbf{x_2} - (\mathbf{x_1'} + \mathbf{x_2'})) + (\mathbf{x_1} - \mathbf{x_2} - (\mathbf{x_1'} + \mathbf{x_2'}))^T \mathbf{M}^*(\mathbf{x_1'} + \mathbf{x_2'})| \\
&\quad \leq \; U(|(\mathbf{x_1} - \mathbf{x_2})^T \mathbf{M}^*(\mathbf{x_1} - \mathbf{x_1'})| + |(\mathbf{x_1} - \mathbf{x_2})^T \mathbf{M}^*(\mathbf{x_2'} - \mathbf{x_2})| \\
&\qquad + |(\mathbf{x_1} - \mathbf{x_1'})^T \mathbf{M}^*(\mathbf{x_1'} + \mathbf{x_2'})| + |(\mathbf{x_2'} - \mathbf{x_2})^T \mathbf{M}^*(\mathbf{x_1'} + \mathbf{x_2'})|) \\
&\quad \leq \; U(\|\mathbf{x_1} - \mathbf{x_2}\|_2 \|\mathbf{M}^*\|_{\mathcal{F}} \|\mathbf{x_1} - \mathbf{x_1'}\|_2 + \|\mathbf{x_1} - \mathbf{x_2}\|_2 \|\mathbf{M}^*\|_{\mathcal{F}} \|\mathbf{x_2'} - \mathbf{x_2}\|_2 \\
&\qquad + \|\mathbf{x_1} - \mathbf{x_1'}\|_2 \|\mathbf{M}^*\|_{\mathcal{F}} \|\mathbf{x_1'} - \mathbf{x_2'}\|_2 + \|\mathbf{x_2'} - \mathbf{x_2}\|_2 \|\mathbf{M}^*\|_{\mathcal{F}} \|\mathbf{x_1'} - \mathbf{x_2'}\|_2) \\
&\quad \leq \; \frac{8UR\gamma g_0}{C}.
\end{aligned}
$$

Hence, the example holds by Theorem 7.11. □

The generalization bound for Example 7.1 derived by Jin et al. (2009) using uniform stability arguments has the same order of convergence. However, their framework cannot be used to establish generalization bounds for recent sparse metric learning approaches (Rosales & Fung, 2006; Qi et al., 2009; Ying et al., 2009; Kunapuli & Shavlik, 2012) because sparse algorithms are known not to be stable (Xu et al., 2012a). The key advantage of robustness over stability is that it can accommodate arbitrary p-norms (or even any regularizer which is bounded below by some p-norm), thanks to the equivalence of norms. To illustrate this, we show the robustness when $\|\mathbf{M}\|$ is the L_1 norm (used in Rosales & Fung, 2006; Qi et al., 2009) which promotes sparsity at the component level, the $L_{2,1}$ norm (used in Ying et al., 2009) which induces group sparsity at the column/row level, and the trace norm (used in Kunapuli & Shavlik, 2012) which induces low-rank matrices.

Example 7.2 (L_1 norm)**.** *Algorithm* (7.7) *with* $\|\mathbf{M}\| = \|\mathbf{M}\|_1$ *is* $(|\mathcal{Y}|\mathcal{N}(\gamma, \mathcal{X}, \|\cdot\|_1), \frac{8UR\gamma g_0}{C})$*-robust.*

Proof. See Appendix B.2.4. □

Example 7.3 ($L_{2,1}$ norm and trace norm)**.** *Algorithm* (7.7) *with* $\|\mathbf{M}\| = \|\mathbf{M}\|_{2,1}$ *or* $\|\mathbf{M}\| = \|\mathbf{M}\|_*$ *is* $(|\mathcal{Y}|\mathcal{N}(\gamma, \mathcal{X}, \|\cdot\|_2), \frac{8UR\gamma g_0}{C})$*-robust.*

Proof. See Appendix B.2.5. □

We have seen that kernelization is a convenient way to learn a nonlinear metric. In the following example, we show robustness for a kernelized formulation.

Example 7.4 (Kernelization)**.** *Consider the kernelized version of Algorithm* (7.7)*:*

$$\min_{\mathbf{M}\succeq 0} \frac{1}{n^2} \sum_{(z_i,z_j)\in\mathcal{P}_\mathcal{T}} g(y_i y_j[1 - d^2_\mathbf{M}(\phi(\mathbf{x_i}), \phi(\mathbf{x_j}))]) \quad + \quad C\|\mathbf{M}\|_\mathbb{H}, \tag{7.8}$$

where $\phi(\cdot)$ is a feature mapping to a kernel space $\mathbb{H}$, $\|\cdot\|_\mathbb{H}$ the norm function of $\mathbb{H}$ and $k(\cdot,\cdot)$ the kernel function. Consider a cover of $\mathcal{X}$ by $\|\cdot\|_2$ ($\mathcal{X}$ being compact) and let

$$f_\mathbb{H}(\gamma) = \max_{\mathbf{a},\mathbf{b}\in\mathcal{X},\|\mathbf{a}-\mathbf{b}\|_2\leq\gamma} K(\mathbf{a},\mathbf{a}) + K(\mathbf{b},\mathbf{b}) - 2K(\mathbf{a},\mathbf{b}) \quad \textit{and} \quad B_\gamma = \max_{x\in\mathcal{X}} \sqrt{K(\mathbf{x},\mathbf{x})}.$$

If the kernel function is continuous, B_γ and $f_\mathbb{H}$ are finite for any $\gamma > 0$ and thus Algorithm (7.8) *is* $(|\mathcal{Y}|\mathcal{N}(\gamma,\mathcal{X},\|\cdot\|_2), \frac{8UB_\gamma\sqrt{f_\mathbb{H}}g_0}{C})$*-robust.*

Proof. See Appendix B.2.6. □

Using triplet-based robustness (7.1), we can for instance show the robustness of two popular triplet-based metric learning approaches (Schultz & Joachims, 2003; Ying et al., 2009) for which no generalization guarantees were known (to the best of our knowledge). Recall that these algorithms have the following form:

$$\min_{\mathbf{M}\succeq 0} \frac{1}{|\mathcal{R}_\mathcal{T}|} \sum_{(z_i,z_j,z_k)\in\mathcal{R}_\mathcal{T}} [1 - d^2_\mathbf{M}(\mathbf{x_i},\mathbf{x_k}) + d^2_\mathbf{M}(\mathbf{x_i},\mathbf{x_j})]_+ \quad + \quad C\|\mathbf{M}\|,$$

where Schultz & Joachims (2003) use $\|\mathbf{M}\| = \|\mathbf{M}\|_\mathcal{F}$ and Ying et al. (2009) use $\|\mathbf{M}\| = \|\mathbf{M}\|_{1,2}$. These methods are $(\mathcal{N}(\gamma,\mathcal{Z},\|\cdot\|_2), \frac{16UR\gamma g_0}{C})$-robust (by using the same proof technique as in Example 7.1 and Example 7.3). The additional factor 2 comes from the use of triplets instead of pairs.

Furthermore, we can easily prove similar results for other forms of metrics using the same technique. For instance, when the function is the bilinear similarity $\mathbf{x_i}^T\mathbf{M}\mathbf{x_j}$ where $\mathbf{M}$ is not constrained to be PSD (see for instance Chechik et al., 2009; Qamar et al., 2008; Bellet et al., 2012c), we can improve the robustness to $2UR\gamma g_0/C$.

7.5 Conclusion

In this chapter, we proposed a new theoretical framework for establishing generalization bounds for metric learning algorithms, based on the notion of algorithmic robustness originally introduced by (Xu & Mannor, 2010, 2012). We showed that robustness can be adapted to pair and triplet-based metric learning and can be used to derive generalization guarantees without assuming that the pairs or triplets are drawn i.i.d. Furthermore, we showed that a weak notion of robustness characterizes the generalizability of metric learning algorithms, justifying that robustness is fundamental for such algorithms. The

proposed framework is used to derive generalization bounds for a large class of metric learning algorithms with different regularizations, such as sparsity-inducing norms, making the analysis more powerful and general than the (few) existing frameworks. Moreover, almost no algorithm-specific argument is needed to derive these bounds.

It is worth noting that our adaptation of robustness to metric learning is relatively straightforward: in most cases, the proof techniques of Xu & Mannor (2010, 2012) could be reused with only slight modification. Nevertheless, this adaptation is promising since it leads to generalization bounds for many metric learning methods that could not be studied through the prism of previous frameworks. Note that it could be used to make the link between the generalization ability of metric learning methods and their (ϵ, γ, τ)-goodness, in a similar fashion to what we did in Chapter 5 with uniform stability. An obvious drawback of the proposed framework is that the resulting bounds are loose and often similar from one method to another due to the use of covering numbers and equivalence of norms.

A natural perspective is to consider different, harder settings. Besides extending our framework to more general loss functions (for example those that use both pairs and triplets, such as Weinberger & Saul, 2009) and regularizers (e.g., the LogDet divergence used in Davis et al., 2007; Jain et al., 2008), studying other paradigms for metric learning (such as unsupervised, semi-supervised or domain adaptation methods) would be of great interest.

Lastly, another interesting avenue is to design a metric learning algorithm that would maximize the robustness of the resulting metric.

CHAPTER 8
Conclusion & Perspectives

In this thesis, we have addressed some important limitations of existing supervised metric learning methods by proposing new approaches for feature vectors and structured data. We paid particular attention to the desirable properties and justifications of each contribution presented in this document. We studied both theoretical frameworks and algorithmic issues, but also the applicability of the different approaches. Overall, it constitutes a wide range of research.

Our first contribution (which was actually not a metric learning algorithm) was to propose a new string kernel built from learned edit similarities. This kernel combines powerful learned edit similarities with the classification performance of support vector machines: it is more adaptable than classic string kernels (such as the spectrum, subsequence or mismatch kernels) while being guaranteed to be PSD, unlike other kernels based on the edit distance. We provided a tractable way to compute it, although the proposed solution can remain computationally expensive.

In order to avoid the cost of transforming learned edit similarities into kernels, we then proposed to use them directly to build a linear classifier, following the framework of learning with (ϵ, γ, τ)-good similarity functions (Balcan & Blum, 2006; Balcan et al., 2008a,b). We observed that this yields competitive results in practice. We went one step further by introducing our main second contribution with GESL, a string and tree edit similarity learning method driven by a relaxation of (ϵ, γ, τ)-goodness. The problem is formulated as an efficient convex quadratic program and solved by convex optimization tools, thereby avoiding the use of expensive and locally optimal EM-based algorithms. Unlike many other edit metric learning methods, we were able to use the information brought by both positive and negative pairs, and to derive generalization guarantees for the learned similarity using uniform stability arguments. These guarantees give an upper bound on the true risk of the classifier built from the learned similarity (although a rather loose one). Furthermore, experimental evaluation showed the accuracy of the method but also its ability to output sparse models, which is a valuable property from a practical point of view. Note that the source code for GESL is available and distributed under GNU/GPL 3 license.[1]

[1] Download from: http://labh-curien.univ-st-etienne.fr/~bellet/

To provide a wider range of applicability, our third contribution was an extension of the ideas of GESL to metric learning from feature vectors. The proposed approach, called SLLC, takes advantage of the simple form of the bilinear similarity to efficiently optimize the actual (ϵ, γ, τ)-goodness, instead of only a loose upper bound in GESL. In this context, the similarity is not learned from local pairs or triplets but according to a global criterion. We also kernelized SLLC to be able to learn linear similarities in a nonlinear feature space induced by a kernel. Generalization guarantees based on uniform stability are established for SLLC and give a tighter bound on the true risk of the linear classifier. To the best of our knowledge, GESL and SLLC are the first metric learning methods for which the link between the quality of the learned metric and the error of the classifier using it is formally established.

Finally, purely on the theoretical side, our last contribution overcame the limitations of the previous frameworks studying the generalization of metric learning algorithms. It is based on a relatively straightforward adaptation of algorithmic robustness (Xu & Mannor, 2010, 2012) but provides an easy way to derive nontrivial results. We illustrated this by showing how it can be used to prove the robustness of a large class of metric learning algorithms, thereby establishing generalization guarantees for methods that could not be handled with previous arguments.

Staying in the scope of the proposed methods, the adaptation to other metrics or other regularizers are possible future directions. In particular, extending the methods to sparsity-inducing regularizers (in order to obtain more interpretable results as well as additional properties such as low-rank solutions and dimensionality reduction) can be done without giving up generalization guarantees, thanks to the theoretical contribution of Chapter 7. To improve the scalability of the approaches, an interesting avenue would be to develop online versions of the algorithms. Another promising idea for future work is to explore the field of information geometry, in particular to study the problem of metric learning in the context of Bregman divergences (Bregman, 1967). Such divergences are known to generalize many metrics for vectors and matrices, and have interesting properties for solving tasks such as clustering (see e.g., Banerjee et al., 2005; Fischer, 2010). To the best of our knowledge, learning Bregman divergences has only been addressed by Wu et al. (2009, 2012).

From a more high-level perspective, many questions remain open as to the theoretical understanding of metric learning. Some of our contributions make the link between the learned metric and its performance in classification, but our results are so far restricted to the context of linear classification, relying on (ϵ, γ, τ)-goodness. A promising avenue would be to derive methods or analytical frameworks capable of making that link for other classifiers. In particular, since most learned metrics are used in k-NN, tying the generalization ability of the learned metric to the true risk of the k-NN classifier would constitute a beautiful result. One could also derive theoretically sound metric learning methods for other supervised learning tasks, such as regression or ranking, using the

recently proposed generalization of the notion of similarity goodness to these settings (Kar & Jain, 2012).

Another interesting perspective would be to study the generalization ability of learned metrics in other settings, such as domain adaptation (Mansour et al., 2009; Ben-David et al., 2010). Domain adaptation (DA) studies the generalization ability of a hypothesis learned from labeled *source* data and used to predict the labels of *target* data, where the distributions generating the source and target data are different. It was shown that successful adaptation is possible when the two distributions are not too different — a common example of such situation is covariate shift, where only the data distributions are different, while the conditional distribution of labels given a data point remains the same (see for instance Bickel et al., 2009, and references therein). Although a few DA metric learning methods already exist (Cao et al., 2011; Geng et al., 2011), insights provided by DA generalization bounds (Mansour et al., 2009; Ben-David et al., 2010) could be used to derive theoretically well-founded approaches.

Finally, one could also focus on clustering, since metrics are essential to many clustering algorithms (such as the prominent K-Means). We identify two promising directions for future research. First, one could use the fact that algorithmic robustness is based on a partition of the input space. This geometric interpretation seems particularly relevant to clustering, and a metric learning algorithm that maximizes a notion of robustness could be appropriate to deal with clustering tasks. Another avenue could consist in formally determining which properties of a metric are important to induce quality clusterings. The work of Balcan et al. (2008c) is a first attempt towards a better understanding of this question.

List of Publications

International Journals

Aurélien Bellet, Amaury Habrard, and Marc Sebban. Good edit similarity learning by loss minimization. *Machine Learning Journal (MLJ)*, 89(1):5–35, 2012b.

Aurélien Bellet, Marc Bernard, Thierry Murgue, and Marc Sebban. Learning state machine-based string edit kernels. *Pattern Recognition (PR)*, 43(6):2330–2339, 2010.

International Conferences

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)*, 2012c.

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, 2011c.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. An Experimental Study on Learning with Good Edit Similarity Functions. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 126–133, 2011a.

French Conferences

Aurélien Bellet, Amaury Habrard, and Marc Sebban. Apprentissage de bonnes similarités pour la classification linéaire parcimonieuse. In *French Conference on Machine Learning (CAp)*, pages 302–317, 2012a.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. Apprentissage Parcimonieux à partir de Fonctions de Similarité d'Édition (ϵ, γ, τ)-Good. In *French Conference on Machine Learning (CAp)*, pages 103–118, 2011b.

Aurélien Bellet, Marc Bernard, Thierry Murgue, and Marc Sebban. Apprentissage de noyaux d'édition de séquences. In *French Conference on Machine Learning (CAp)*, pages 93–108, 2009. Best paper award.

APPENDIX A
Learning Conditional Edit Probabilities

Our string edit kernel introduced in Chapter 4 is based on edit probabilities learned from a generative or discriminative probabilistic model. In the experimental section, we build the kernel from the method of Oncina & Sebban (2006), which is based on estimating the parameters of a conditional memoryless transducer. This appendix gives the technical details of their approach.

Recall that $\mathcal{S}$ denotes the set of positive pairs. For the sake of simplicity, we assume that the input and the output alphabets are the same, denoted by Σ. In the following, unless stated otherwise, symbols are denoted by $\mathsf{a}, \mathsf{b}, \ldots$, and pairs of input and output strings by $(\mathsf{x}, \mathsf{x}')$ or $(\mathsf{w}, \mathsf{w}')$ when needed. Let f be a function such that $[f(\mathsf{x})]_{\pi(\mathsf{x},\ldots)}$ is equal to $f(\mathsf{x})$ if the predicate $\pi(x, \ldots)$ holds and 0 otherwise, where x is a (set of) dummy variable(s). In this appendix, for notational convenience, we will see the edit probability matrix as a function. Let c be the conditional probability function that returns for any edit operation $(\mathsf{b}|\mathsf{a})$ the probability to output the symbol b given an input symbol a.

The aim of this appendix is to show how one can automatically learn the function c from the training pairs $\mathcal{S}$. The values $c(\mathsf{b}|\mathsf{a}), \forall \mathsf{a} \in \Sigma \cup \{\$\}, \mathsf{b} \in \Sigma \cup \{\$\}$ represent the parameters of the memoryless machine T. These parameters are trained using an EM-based algorithm that relies on the so-called forward and backward functions.

The conditional edit probability $p_e(\mathsf{x}'|\mathsf{x})$ of the string x' given an input string x can be recursively computed using the forward function $\alpha : \Sigma^* \times \Sigma^* \to \mathbb{R}_+$ defined as follows:

$$\begin{aligned}
\alpha(\mathsf{x}'|\mathsf{x}) \quad = \quad & [1]_{\mathsf{x}=\$\wedge\mathsf{x}'=\$} \\
& + [c(\mathsf{b}|\mathsf{a}) \cdot \alpha(\mathsf{w}'|\mathsf{w})]_{\mathsf{x}=\mathsf{wa}\wedge\mathsf{x}'=\mathsf{w}'\mathsf{b}} \\
& + [c(\$|\mathsf{a}) \cdot \alpha(\mathsf{x}'|\mathsf{w}]_{\mathsf{x}=\mathsf{wa}} \\
& + [c(\mathsf{b}|\$) \cdot \alpha(\mathsf{w}'|\mathsf{x})]_{\mathsf{x}'=\mathsf{w}'\mathsf{b}}.
\end{aligned}$$

Using $\alpha(\mathsf{x}'|\mathsf{x})$, we get

$$p_e(\mathsf{x}'|\mathsf{x}) = c(\$|\$) \cdot \alpha(\mathsf{x}'|\mathsf{x}),$$

where $c(\$|\$)$ is the probability of the termination symbol of a string.

In a symmetric way, $p_e(\mathsf{x}'|\mathsf{x})$ can be recursively computed using the backward function $\beta : \Sigma^* \times \Sigma^* \to \mathbb{R}_+$ defined as follows:

$$\begin{aligned}\beta(\mathsf{x}'|\mathsf{x}) \quad = \quad & [1]_{\mathsf{x}=\$\wedge\mathsf{x}'=\$} \\ & + [c(\mathsf{b}|\mathsf{a}) \cdot \beta(\mathsf{w}'|\mathsf{w})]_{\mathsf{x}=\mathsf{aw}\wedge\mathsf{x}'=\mathsf{bw}'} \\ & + [c(\$|\mathsf{a}) \cdot \beta(\mathsf{x}'|\mathsf{w})]_{\mathsf{x}=\mathsf{aw}} \\ & + [c(\mathsf{b}|\$) \cdot \beta(\mathsf{w}'|\mathsf{x})]_{\mathsf{x}'=\mathsf{bw}'}.\end{aligned}$$

And we get that

$$p_e(\mathsf{x}'|\mathsf{x}) = c(\$|\$) \cdot \beta(\mathsf{x}'|\mathsf{x}).$$

Both functions can be computed in $O(|\mathsf{x}||\mathsf{x}'|)$ time using a dynamic programming technique and will be used in the following to learn the function c.

In the considered model, a probability distribution is assigned conditionally to each input string, i.e.,

$$\sum_{\mathsf{x}'\in\Sigma^*} p(\mathsf{x}'|\mathsf{x}) \in \{0,1\} \quad \forall \mathsf{x} \in \Sigma^*.$$

This is equal to 0 when the input string x is not in the domain of the function.[1]

It can be shown (see Oncina & Sebban, 2006, for the proof) that correct normalization of each conditional distribution is obtained when the following conditions over the function c are fulfilled:

$$\begin{aligned} c(\$|\$) &> 0, \\ c(\mathsf{b}|\mathsf{a}), c(\mathsf{b}|\$), c(\$|\mathsf{a}) &\geq 0, \quad \forall \mathsf{a} \in \Sigma, \mathsf{b} \in \Sigma, \\ \sum_{\mathsf{b}\in\Sigma} c(\mathsf{b}|\$) + \sum_{\mathsf{b}\in\Sigma} c(\mathsf{b}|\mathsf{a}) + c(\$|\mathsf{a}) &= 1, \quad \forall \mathsf{a} \in \Sigma, \\ \sum_{\mathsf{b}\in\Sigma} c(\mathsf{b}|\$) + c(\$|\$) &= 1. \end{aligned}$$

The EM algorithm (Dempster et al., 1977) can be used in order the find the optimal parameters of the function c by alternating between an E-step and an M-step. Given an auxiliary $(|\Sigma|+1) \times (|\Sigma|+1)$ matrix δ, the E-step aims at computing the values of δ as follows: $\forall \mathsf{a} \in \Sigma, \mathsf{b} \in \Sigma$,

[1] If $p_e(\mathsf{x}) = 0$ then $p_e(\mathsf{x}, \mathsf{x}') = 0$ and as $p_e(\mathsf{x}'|\mathsf{x}) = \frac{p_e(\mathsf{x},\mathsf{x}')}{p(\mathsf{x})}$ we have a $\frac{0}{0}$ indeterminate. We choose to avoid it by taking $\frac{0}{0} = 0$, in order to keep $\sum_{\mathsf{x}'\in\Sigma^*} p(\mathsf{x}'|\mathsf{x})$ finite.

$$
\begin{aligned}
\delta(\mathsf{b}|\mathsf{a}) &= \sum_{(\mathsf{xaw},\mathsf{x'bw'})\in\mathcal{S}} \frac{\alpha(\mathsf{x'}|\mathsf{x})\cdot c(\mathsf{b}|\mathsf{a})\cdot\beta(\mathsf{w'}|\mathsf{w})\cdot c(\$|\$)}{p_e(\mathsf{x'bw'}|\mathsf{xaw})},\\
\delta(\mathsf{b}|\$) &= \sum_{(\mathsf{xw},\mathsf{x'bw'})\in\mathcal{S}} \frac{\alpha(\mathsf{x'}|\mathsf{x})\cdot c(\mathsf{b}|\$)\cdot\beta(\mathsf{w'}|\mathsf{w})\cdot c(\$|\$)}{p_e(\mathsf{x'bw'}|\mathsf{xw})},\\
\delta(\$|\mathsf{a}) &= \sum_{(\mathsf{xaw},\mathsf{x'w'})\in\mathcal{S}} \frac{\alpha(\mathsf{x'}|\mathsf{x})c(\$|\mathsf{a})\cdot\beta(\mathsf{w'}|\mathsf{w})\cdot c(\$|\$)}{p_e(\mathsf{x'w'}|\mathsf{xaw})},\\
\delta(\$|\$) &= \sum_{(\mathsf{x},\mathsf{x'})\in\mathcal{S}} \frac{\alpha(\mathsf{x'}|\mathsf{x})\cdot c(\$|\$)}{p_e(\mathsf{x'}|\mathsf{x})} = |\mathcal{S}|.
\end{aligned}
$$

The M-step allows us to get the current edit costs:

$$
\begin{aligned}
c(\mathsf{b}|\$) &= \frac{\delta(\mathsf{b}|\$)}{N}, && \text{(insertion)}\\
c(\$|\$) &= \frac{N-N(\$)}{N}, && \text{(termination symbol)}\\
c(\mathsf{b}|\mathsf{a}) &= \frac{\delta(\mathsf{b}|\mathsf{a})}{N(\mathsf{a})}\cdot\frac{N-N(\$)}{N}, && \text{(substitution)}\\
c(\$|\mathsf{a}) &= \frac{\delta(\$|\mathsf{a})}{N(\mathsf{a})}\cdot\frac{N-N(\$)}{N}, && \text{(deletion)}
\end{aligned}
$$

where

$$
N = \sum_{\substack{a\in\Sigma\cup\{\$\}\\ b\in\Sigma\cup\{\$\}}} \delta(b|a), \qquad N(\$) = \sum_{b\in\Sigma}\delta(b|\$), \qquad N(a) = \sum_{b\in\Sigma\cup\{\$\}}\delta(b|a).
$$

APPENDIX B
Proofs

B.1 Proofs of Chapter 5

B.1.1 Proof of Lemma 5.8

Lemma Let $F_{\mathcal{T}}$ and $F_{\mathcal{T}^{i,z}}$ be the functions to optimize, $\mathbf{C}_{\mathcal{T}}$ and $\mathbf{C}_{\mathcal{T}^{i,z}}$ their corresponding minimizers, and β the regularization parameter used in $GESL_L$. Let $\Delta\mathbf{C} = (\mathbf{C}_{\mathcal{T}} - \mathbf{C}_{\mathcal{T}^{i,z}})$. For any $t \in [0,1]$:

$$\|\mathbf{C}_{\mathcal{T}}\|_{\mathcal{F}}^2 - \|\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}\|_{\mathcal{F}}^2 + \|\mathbf{C}_{\mathcal{T}^{i,z}}\|_{\mathcal{F}}^2 - \|\mathbf{C}_{\mathcal{T}^{i,z}} + t\Delta\mathbf{C}\|_{\mathcal{F}}^2 \leq \frac{(2n_{\mathcal{T}} + n_{\mathcal{L}})t2k}{\beta n_{\mathcal{T}} n_{\mathcal{L}}}\|\Delta\mathbf{C}\|_{\mathcal{F}}.$$

Proof. The first steps of this proof are similar to the proof of Lemma 20 in (Bousquet & Elisseeff, 2002) which we recall for the sake of completeness. Recall that any convex function g verifies

$$\forall x, y, \forall t \in [0,1], g(x + t(y - x)) - g(x) \leq t(g(y) - g(x)).$$

$R^{\ell}_{\mathcal{T}^{i,z}}$ is convex and thus for any $t \in [0,1]$,

$$R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}) - R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}}) \leq t(R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}) - R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}})). \tag{B.1}$$

Switching the role of $\mathbf{C}_{\mathcal{T}}$ and $\mathbf{C}_{\mathcal{T}^{i,z}}$, we get:

$$R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}} + t\Delta\mathbf{C}) - R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}) \leq t(R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}}) - R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}})). \tag{B.2}$$

Summing up inequalities (B.1) and (B.2) yields

$$R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}) - R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}}) + R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}} + t\Delta\mathbf{C}) - R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}) \leq 0. \tag{B.3}$$

Now, since $\mathbf{C}_{\mathcal{T}}$ and $\mathbf{C}_{\mathcal{T}^{i,z}}$ are minimizers of $F_{\mathcal{T}}$ and $F_{\mathcal{T}^{i,z}}$ respectively, we have:

$$F_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}}) - F_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}) \leq 0 \tag{B.4}$$

$$F_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}) - F_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}} + t\Delta\mathbf{C}) \leq 0. \tag{B.5}$$

By summing up (B.4) and (B.5) we get:

$$\begin{aligned} R^\ell_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}}) + \beta\|\mathbf{C}_{\mathcal{T}}\|_{\mathcal{F}} - \Big(R^\ell_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}) + \beta\|\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}\|_{\mathcal{F}}\Big) + \\ R^\ell_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}) + \beta\|\mathbf{C}_{\mathcal{T}^{i,z}}\|_{\mathcal{F}} - (R^\ell_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}} + t\Delta\mathbf{C}) + \beta\|\mathbf{C}_{\mathcal{T}^{i,z}} + t\Delta\mathbf{C}\|_{\mathcal{F}}) \quad \leq \quad 0. \end{aligned}$$

By summing this last inequality with (B.3), we obtain

$$\begin{aligned} R^\ell_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}}) + \beta\|\mathbf{C}_{\mathcal{T}}\|_{\mathcal{F}} - \Big(R^\ell_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}) + \beta\|\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}\|_{\mathcal{F}}\Big) + \\ \beta\|\mathbf{C}_{\mathcal{T}^{i,z}}\|_{\mathcal{F}} - (\beta\|\mathbf{C}_{\mathcal{T}^{i,z}} + t\Delta\mathbf{C}\|_{\mathcal{F}}) + R^\ell_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}) - R^\ell_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}}) \quad \leq \quad 0. \end{aligned}$$

Let $B = R^\ell_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}) - R^\ell_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}) - (R^\ell_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}}) - R^\ell_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}}))$, we have then

$$\beta(\|\mathbf{C}_{\mathcal{T}}\|_{\mathcal{F}} - \|\mathbf{C}_{\mathcal{T}} - t(\Delta\mathbf{C})\|_{\mathcal{F}} + \|\mathbf{C}_{\mathcal{T}^{i,z}}\|_{\mathcal{F}} - \|\mathbf{C}_{\mathcal{T}^{i,z}} + t(\Delta\mathbf{C})\|_{\mathcal{F}}) \leq B. \tag{B.6}$$

We now derive a bound for B. In the following, $z'_{k_j} \in \mathcal{T}$ denotes the jth landmark associated to $z_k \in \mathcal{T}$ such that $f_{land_{\mathcal{T}}}(z_k, z'_{k_j}) = 1$ in $\mathcal{T}$, and $z'^i_{k_j} \in \mathcal{T}^{i,z}$ the jth landmark associated to $z^i_k \in \mathcal{T}^{i,z}$ such that $f_{land_{\mathcal{T}^{i,z}}}(z^i_k, z'^i_{k_j}) = 1$ in $\mathcal{T}^{i,z}$.

$$\begin{aligned} B \quad &\leq \quad |R^\ell_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}) - R^\ell_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}) - (R^\ell_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}}) - R^\ell_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}}))| \\ &\leq \quad \frac{1}{n_{\mathcal{T}} n_{\mathcal{L}}} \Bigg| \sum_{k=1}^{n_{\mathcal{T}}} \sum_{j=1}^{n_{\mathcal{L}}} \ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_k, z'_{k_j}) - \sum_{k=1}^{n_{\mathcal{T}}} \sum_{j=1}^{n_{\mathcal{L}}} \ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z^i_k, z'^i_{k_j}) \\ &\qquad - \left(\sum_{k=1}^{n_{\mathcal{T}}} \sum_{j=1}^{n_{\mathcal{L}}} \ell(\mathbf{C}_{\mathcal{T}}, z_k, z'_{k_j}) - \sum_{k=1}^{n_{\mathcal{T}}} \sum_{j=1}^{n_{\mathcal{L}}} \ell(\mathbf{C}_{\mathcal{T}}, z^i_k, z'^i_{k_j}) \right) \Bigg| \\ &\leq \quad \frac{1}{n_{\mathcal{T}} n_{\mathcal{L}}} \Bigg| \sum_{j=1}^{n_{\mathcal{L}}} \Big(\ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_i, z'_{i_j}) - \ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z, z'^i_j) \Big) + \\ &\qquad \sum_{\substack{k=1 \\ k\neq i}}^{n_{\mathcal{T}}} \sum_{j=1}^{n_{\mathcal{L}}} \Big(\ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_k, z'_{k_j}) - \ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z^i_k, z'^i_{k_j}) \Big) \\ &\qquad - \left(\sum_{k=1}^{n_{\mathcal{T}}} \sum_{j=1}^{n_{\mathcal{L}}} \ell(\mathbf{C}_{\mathcal{T}}, z_k, z'_{k_j}) - \sum_{k=1}^{n_{\mathcal{T}}} \sum_{j=1}^{n_{\mathcal{L}}} \ell(\mathbf{C}_{\mathcal{T}}, z^i_k, z'^i_{k_j}) \right) \Bigg| \end{aligned}$$

This inequality is obtained by developing the sum of the first two terms of the second line. The examples z_i in $\mathcal{T}$ and z in $\mathcal{T}^{i,z}$ have $n_{\mathcal{L}}$ landmarks defined by $f_{land_{\mathcal{T}}}$ and $f_{land_{\mathcal{T}^{i,z}}}$ respectively.
Note that the samples of $n_{\mathcal{T}} - 1$ elements $\mathcal{T}\backslash\{z_i\}$ and $\mathcal{T}^{i,z}\backslash\{z\}$ are the same and thus $z_k = z^i_k$ when $k \neq i$. Therefore, for any $z_k \in \mathcal{T}\backslash\{z_i\}$, the sets of landmarks $\mathcal{L}^{z_k}_{\mathcal{T}} = \{z'_{k_j} \in \mathcal{T} | f_{land_{\mathcal{T}}}(z_k, z'_{k_j}) = 1\}$ and $\mathcal{L}^{z_k}_{\mathcal{T}^{i,z}} = \{z'^i_{k_j} \in \mathcal{T}^{i,z} | f_{land_{\mathcal{T}^{i,z}}}(z_k, z'^i_{k_j}) = 1\}$ differ on at most two elements, say $z_i, z'_{k_{j_2}} \in \mathcal{L}^{z_k}_{\mathcal{T}}\backslash\mathcal{L}^{z_k}_{\mathcal{T}^{i,z}}$ and $z, z'^i_{k_{j_1}} \in \mathcal{L}^{z_k}_{\mathcal{T}^{i,z}}\backslash\mathcal{L}^{z_k}_{\mathcal{T}}$. Thus, some terms cancel

out and we have:

$$\begin{aligned} B \leq \frac{1}{n_{\mathcal{T}} n_{\mathcal{L}}} \Bigg| & \sum_{j=1}^{n_{\mathcal{L}}} \Big(\ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_i, z'_{i_j}) - \ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z, z'^{i}_{j}) \Big) + \sum_{\substack{k=1 \\ k \neq i}}^{n_{\mathcal{T}}} \Big(\ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_k, z_i) \\ & - \ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_k, z'^{i}_{k_{j_1}}) + \ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_k, z'_{k_{j_2}}) - \ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_k, z) \Big) \\ & - \left(\sum_{k=1}^{n_{\mathcal{T}}} \sum_{j=1}^{n_{\mathcal{L}}} \ell(\mathbf{C}_{\mathcal{T}}, z_k, z'_{k_j}) - \sum_{k=1}^{n_{\mathcal{T}}} \sum_{j=1}^{n_{\mathcal{L}}} \ell(\mathbf{C}_{\mathcal{T}}, z_k, z'^{i}_{k_j}) \right) \Bigg| \end{aligned}$$

The first two lines of the absolute value can be bounded by:

$$(2(n_{\mathcal{T}} - 1) + n_{\mathcal{L}}) \sup_{\substack{z_1, z_2 \in T \\ z_3, z_4 \in \mathcal{T}^{i,z}}} |\ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_1, z_2) - \ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_3, z_4)|.$$

The same analysis can be done for the part in parentheses of the last line of the absolute value and we can take the pair of examples in $\mathcal{T}$ and in $\mathcal{T}^{i,z}$ maximizing the whole absolute value to obtain the next inequality:

$$\begin{aligned} B \leq \frac{2(n_{\mathcal{T}} - 1) + n_{\mathcal{L}}}{n_{\mathcal{T}} n_{\mathcal{L}}} \sup_{\substack{z_1, z_2 \in T \\ z_3, z_4 \in \mathcal{T}^{i,z}}} & |\ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_1, z_2) - \ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_3, z_4) \\ & - (\ell(\mathbf{C}_{\mathcal{T}}, z_1, z_2) - \ell(\mathbf{C}_{\mathcal{T}}, z_3, z_4))| \,. \end{aligned}$$

We continue by applying a reordering of the terms and the triangular inequality to get the next result:

$$\begin{aligned} B \leq \frac{2(n_{\mathcal{T}} - 1) + n_{\mathcal{L}}}{n_{\mathcal{T}} n_{\mathcal{L}}} \Bigg(& \sup_{z_1, z_2 \in T} |\ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_1, z_2) - \ell(\mathbf{C}_{\mathcal{T}}, z_1, z_2)| + \\ & \sup_{z_3, z_4 \in \mathcal{T}^{i,z}} |\ell(\mathbf{C}_{\mathcal{T}} - t\Delta\mathbf{C}, z_3, z_4) - \ell(\mathbf{C}_{\mathcal{T}}, z_3, z_4)| \Bigg) \,. \end{aligned}$$

We then use twice the k-lipschitz property of ℓ which leads to:

$$\begin{aligned} B &\leq \frac{(2n_{\mathcal{T}} + n_{\mathcal{L}})}{n_{\mathcal{T}} n_{\mathcal{L}}} 2k \| - t\Delta\mathbf{C}\|_{\mathcal{F}} \\ &\leq \frac{(2n_{\mathcal{T}} + n_{\mathcal{L}})}{n_{\mathcal{T}} n_{\mathcal{L}}} t2k \|\Delta\mathbf{C}\|_{\mathcal{F}}. \end{aligned}$$

Then, by applying this bound on B from inequality (B.6), we get the lemma. □

B.1.2 Proof of Lemma 5.11

Lemma For any learning method of estimation error $D_{\mathcal{T}}$ and satisfying a uniform stability in $\frac{\kappa}{n_{\mathcal{T}}}$, we have $\mathbb{E}_{\mathcal{T}}[D_{\mathcal{T}}] \leq \frac{2\kappa}{n_{\mathcal{T}}}$.

Proof. First recall that for any T, z, z', by hypothesis of uniform stability we have:

$$|\ell(\mathbf{C}_{\mathcal{T}}, z, z') - \ell(\mathbf{C}_{\mathcal{T}^{k,z}}, z, z')| \leq \sup_{z_1,z_2} |\ell(\mathbf{C}_{\mathcal{T}}, z_1, z_2) - \ell(\mathbf{C}_{\mathcal{T}^{k,z}}, z_1, z_2)| \leq \frac{\kappa}{n_{\mathcal{T}}}.$$

Now, we can derive a bound for $\mathbb{E}_{\mathcal{T}}[D_{\mathcal{T}}]$.

$$\begin{aligned}
\mathbb{E}_{\mathcal{T}}[D_{\mathcal{T}}] &\leq \mathbb{E}_{\mathcal{T}}[\mathbb{E}_{z,z'}[\ell(\mathbf{C}_{\mathcal{T}}, z, z')] - R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}})] \\
&\leq \mathbb{E}_{\mathcal{T},z,z'}[|\ell(\mathbf{C}_{\mathcal{T}}, z, z') - \frac{1}{n_{\mathcal{T}}}\sum_{k=1}^{n_{\mathcal{T}}}\frac{1}{n_{\mathcal{L}}}\sum_{j=1}^{n_{\mathcal{L}}}\ell(\mathbf{C}_{\mathcal{T}}, z_k, z'_{k_j})|] \\
&\leq \mathbb{E}_{\mathcal{T},z,z'}[|\frac{1}{n_{\mathcal{T}}}\sum_{k=1}^{n_{\mathcal{T}}}\frac{1}{n_{\mathcal{L}}}\sum_{j=1}^{n_{\mathcal{L}}}(\ell(\mathbf{C}_{\mathcal{T}}, z, z') - \ell(\mathbf{C}_{\mathcal{T}^{k,z}}, z_k, z'_{k_j}) + \\
&\qquad \ell(\mathbf{C}_{\mathcal{T}^{k,z}}, z_k, z'_{k_j}) - \ell(\mathbf{C}_{\mathcal{T}}, z_k, z'_{k_j}))|] \\
&\leq \mathbb{E}_{\mathcal{T},z,z'}[|\frac{1}{n_{\mathcal{T}}}\sum_{k=1}^{n_{\mathcal{T}}}\frac{1}{n_{\mathcal{L}}}\sum_{j=1}^{n_{\mathcal{L}}}(\ell(\mathbf{C}_{\mathcal{T}}, z, z') - \ell(\mathbf{C}_{\mathcal{T}^{k,z}}, z_k, z'_{k_j}))|] + \\
&\quad \frac{1}{n_{\mathcal{T}}}\sum_{k=1}^{n_{\mathcal{T}}}\frac{1}{n_{\mathcal{L}}}\sum_{j=1}^{n_{\mathcal{L}}}\mathbb{E}_{\mathcal{T},z,z'}[|\ell(\mathbf{C}_{\mathcal{T}^{k,z}}, z_k, z'_{k_j}) - \ell(\mathbf{C}_{\mathcal{T}}, z_k, z'_{k_j})|] \\
&\leq \mathbb{E}_{\mathcal{T},z,z'}[|\frac{1}{n_{\mathcal{T}}}\sum_{k=1}^{n_{\mathcal{T}}}\frac{1}{n_{\mathcal{L}}}\sum_{j=1}^{n_{\mathcal{L}}}(\ell(\mathbf{C}_{\mathcal{T}}, z, z') - \ell(\mathbf{C}_{\mathcal{T}^{k,z}}, z_k, z'_{k_j}))|] + \frac{\kappa}{n_{\mathcal{T}}}.
\end{aligned}$$

The last inequality is obtained by applying the hypothesis of uniform stability to the second part of the sum. Now, since $\mathcal{T}$, z and z' are drawn i.i.d. from distribution P, we do not change the expected value by replacing one point with another and thus:

$$\mathbb{E}_{\mathcal{T},z,z'}[|\ell(\mathbf{C}_{\mathcal{T}}, z, z') - \ell(\mathbf{C}_{\mathcal{T}}, z_k, z')|] = \mathbb{E}_{\mathcal{T},z,z'}[|\ell(\mathbf{C}^{z,k}_{\mathcal{T}}, z_k, z') - \ell(\mathbf{C}_{\mathcal{T}}, z_k, z')|].$$

Then, by applying this trick twice on the first element of the sum:

$$\begin{aligned}
\mathbb{E}_{\mathcal{T}}[D_{\mathcal{T}}] &\leq \mathbb{E}_{\mathcal{T},z,z'}[|\frac{1}{n_{\mathcal{T}}}\sum_{k=1}^{n_{\mathcal{T}}}\frac{1}{n_{\mathcal{L}}}\sum_{j=1}^{n_{\mathcal{L}}}(\ell(\mathbf{C}_{\mathcal{T}^{k,z}}, z_k, z') - \ell(\mathbf{C}_{\mathcal{T}^{k,z}}, z_k, z'_{k_j}))|] + \frac{\kappa}{n_{\mathcal{T}}} \\
&\leq \mathbb{E}_{\mathcal{T},z,z'}[|\frac{1}{n_{\mathcal{T}}}\sum_{k=1}^{n_{\mathcal{T}}}\frac{1}{n_{\mathcal{L}}}\sum_{j=1}^{n_{\mathcal{L}}}(\ell(\mathbf{C}_{\{T^{k,z}\}^{k_j,z'}}, z_k, z'_{k_j}) - \ell(\mathbf{C}_{\mathcal{T}^{k,z}}, z_k, z'_{k_j}))|] + \frac{\kappa}{n_{\mathcal{T}}} \\
&\leq \frac{\kappa}{n_{\mathcal{T}}} + \frac{\kappa}{n_{\mathcal{T}}},
\end{aligned}$$

which gives the lemma. □

B.1.3 Proof of Lemma 5.12

Lemma For any edit cost matrix learned by $GESL_L$ using $n_{\mathcal{T}}$ training examples and $n_{\mathcal{L}}$ landmarks, and any loss function ℓ satisfying (σ, m)-admissibility, we have the following bound:

$$\forall i, 1 \leq i \leq n_{\mathcal{T}}, \quad \forall z, \quad |D_{\mathcal{T}} - D_{\mathcal{T}^{i,z}}| \leq \frac{2\kappa}{n_{\mathcal{T}}} + \frac{(2n_{\mathcal{T}} + n_{\mathcal{L}})(2\sigma + m)}{n_{\mathcal{T}} n_{\mathcal{L}}}.$$

Proof. First, we derive a bound on $|D_{\mathcal{T}} - D_{\mathcal{T}^{i,z}}|$.

$$\begin{aligned}
&|D_{\mathcal{T}} - D_{\mathcal{T}^{i,z}}| \\
&\quad = |R^{\ell}(\mathbf{C}_{\mathcal{T}}) - R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}}) - (R^{\ell}(\mathbf{C}_{\mathcal{T}^{i,z}}) - R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}))| \\
&\quad = |R^{\ell}(\mathbf{C}_{\mathcal{T}}) - R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}}) - R^{\ell}(\mathbf{C}_{\mathcal{T}^{i,z}}) + R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}) + R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}^{i,z}}) - R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}^{i,z}})| \\
&\quad = |R^{\ell}(\mathbf{C}_{\mathcal{T}}) - R^{\ell}(\mathbf{C}_{\mathcal{T}^{i,z}}) + R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}^{i,z}}) - R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}}) + R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}) - R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}^{i,z}})| \\
&\quad \leq |R^{\ell}(\mathbf{C}_{\mathcal{T}}) - R^{\ell}(\mathbf{C}_{\mathcal{T}^{i,z}})| + |R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}^{i,z}}) - R^{\ell}_{\mathcal{T}}(\mathbf{C}_{\mathcal{T}})| + |R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}) - R^{\ell}_{T}(\mathbf{C}_{\mathcal{T}^{i,z}})| \\
&\quad \leq \mathbb{E}_{z_1,z_2}[|\ell(\mathbf{C}_{\mathcal{T}}, z_1, z_2) - \ell(\mathbf{C}_{\mathcal{T}^{i,z}}, z_1, z_2)|] + \\
&\quad \frac{1}{n_{\mathcal{T}}} \sum_{k=1}^{n_{\mathcal{T}}} \frac{1}{n_{\mathcal{L}}} \sum_{j=1}^{n_{\mathcal{L}}} |\ell(\mathbf{C}_{\mathcal{T}^{i,z}}, z_k, z'_{k_j}) - \ell(\mathbf{C}_{\mathcal{T}}, z_k, z'_{k_j})| + |R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}) - R^{\ell}_{T}(\mathbf{C}_{\mathcal{T}^{i,z}})| \\
&\quad \leq 2\frac{\kappa}{n_{\mathcal{T}}} + |R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}) - R^{\ell}_{T}(\mathbf{C}_{\mathcal{T}^{i,z}})| \text{ by using the hypothesis of stability twice.}
\end{aligned}$$

Now, proving Lemma 5.12 boils down to bounding the last term above. Using arguments similar to those used in the second part of the proof of Lemma 5.8, we get

$$|R^{\ell}_{\mathcal{T}^{i,z}}(\mathbf{C}_{\mathcal{T}^{i,z}}) - R^{\ell}_{T}(\mathbf{C}_{\mathcal{T}^{i,z}})| \leq \frac{(2n_{\mathcal{T}} + n_{\mathcal{L}})}{n_{\mathcal{T}} n_{\mathcal{L}}} \sup_{\substack{z_1,z_2 \in T \\ z_3,z_4 \in \mathcal{T}^{i,z}}} |\ell(\mathbf{C}_{\mathcal{T}^{i,z}}, z_1, z_2) - \ell(\mathbf{C}_{\mathcal{T}^{i,z}}, z_3, z_4)|.$$

Now by the (σ, m)-admissibility of ℓ, we have that:

$$|\ell(\mathbf{C}_{\mathcal{T}^{i,z}}, z_1, z_2) - \ell(\mathbf{C}_{\mathcal{T}^{i,z}}, z_3, z_4)| \leq \sigma|y_1 y_2 - y_3 y_4| + m \leq 2\sigma + m,$$

since whatever the labels, $|y_1 y_2 - y_3 y_4| \leq 2$. This leads us to the desired result. □

B.1.4 Proof of Lemma 5.14

Lemma The function ℓ_{HL} is k-lipschitz with $k = W$.

Proof. We need to bound $|\ell_{HL}(\mathbf{C}, z, z') - \ell_{HL}(\mathbf{C}', z, z')|$ which implies to consider two cases: when z and z' have the same labels and when they have different labels. We

consider here the first case, the second one can be easily derived from the first one (B_1 playing the same role as B_2).

$$\begin{aligned}
|\ell_{HL}(\mathbf{C}, z, z') - \ell_{HL}(\mathbf{C}', z, z')| &\leq |[\sum_{l,c} C_{l,c}\#_{l,c}(\mathsf{x}, \mathsf{x}') - B_2]_+ - [\sum_{l,c} C'_{l,c}\#_{l,c}(\mathsf{x}, \mathsf{x}') - B_2]_+| \\
&\leq |\sum_{l,c} C_{l,c}\#_{l,c}(\mathsf{x}, \mathsf{x}') - B_2 - (\sum_{l,c} C'_{l,c}\#_{l,c}(\mathsf{x}, \mathsf{x}') - B_2)| \\
&\leq |\sum_{l,c} (C_{l,c} - C'_{l,c})\#_{l,c}(\mathsf{x}, \mathsf{x}')| \\
&\leq \|\mathbf{C} - \mathbf{C}'\|_{\mathcal{F}} \|\#(\mathsf{x}, \mathsf{x}')\|_{\mathcal{F}} \\
&\leq W\|\mathbf{C} - \mathbf{C}'\|_{\mathcal{F}}.
\end{aligned}$$

The second line is obtained by the 1-lipschitz property of the hinge loss:

$$|[U]_+ - [V]_+| \leq |U - V|.$$

The fourth one comes from the Cauchy-Schwartz inequality:

$$|\sum_{i=1}^{n}\sum_{j=1}^{m} A_{i,j}B_{i,j}| \leq \|\mathbf{A}\|_{\mathcal{F}}\|\mathbf{B}\|_{\mathcal{F}}.$$

Finally, since by hypothesis $\|\#(z, z')\|_{\mathcal{F}} \leq W$, the lemma holds. □

B.1.5 Proof of Lemma 5.15

Lemma Let $(\mathbf{C}_{\mathcal{T}}, B_1, B_2)$ an optimal solution learned by $GESL_{HL}$ from a training sample $\mathcal{T}$, and let $B_\gamma = max(\eta_\gamma, -log(1/2))$. Then $\|\mathbf{C}_{\mathcal{T}}\|_{\mathcal{F}} \leq \sqrt{\frac{B_\gamma}{\beta}}$.

Proof. Since $(\mathbf{C}_{\mathcal{T}}, B_1, B_2)$ is an optimal solution, the value reached by the objective function is lower than the one obtained with $(\mathbf{0}, B_\gamma, 0)$, where $\mathbf{0}$ denotes the matrix of zeros:

$$\frac{1}{n_{\mathcal{T}}}\sum_{k=1}^{n_{\mathcal{T}}}\frac{1}{n_{\mathcal{L}}}\sum_{j=1}^{n_{\mathcal{L}}}\ell_{HL}(\mathbf{C}, z_k, z'_{k_j}) + \beta\|\mathbf{C}_{\mathcal{T}}\|^2_{\mathcal{F}} \leq \frac{1}{n_{\mathcal{T}}}\sum_{k=1}^{n_{\mathcal{T}}}\frac{1}{n_{\mathcal{L}}}\sum_{j=1}^{n_{\mathcal{L}}}\ell_{HL}(\mathbf{0}, z_k, z'_{k_j}) + \beta\|\mathbf{0}\|^2_{\mathcal{F}} \leq B_\gamma.$$

For the last inequality, note that regardless of the labels of z_k and z'_{k_j}, $\ell_{HL}(\mathbf{0}, z_k, z'_{k_j})$ is bounded either by B_γ or 0. Since

$$\frac{1}{n_{\mathcal{T}}}\sum_{k=1}^{n_{\mathcal{T}}}\frac{1}{n_{\mathcal{L}}}\sum_{j=1}^{n_{\mathcal{L}}}\ell_{HL}(\mathbf{C}, z_k, z'_{k_j}) \geq 0,$$

we get $\beta\|\mathbf{C}_{\mathcal{T}}\|^2_{\mathcal{F}} \leq B_\gamma$. □

B.2 Proofs of Chapter 7

B.2.1 Proof of Theorem 7.5 (pseudo-robustness)

Theorem If a learning algorithm $\mathcal{A}$ is $(K, \epsilon(\cdot), \hat{p}_n(\cdot))$ pseudo-robust and the training pairs $\mathcal{P}_\mathcal{T}$ come from a sample generated by n i.i.d. draws from P, then for any $\delta > 0$, with probability at least $1-\delta$ we have:

$$|R^\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}) - R^\ell_{\mathcal{P}_\mathcal{T}}(\mathcal{A}_{\mathcal{P}_\mathcal{T}})| \leq \frac{\hat{p}_n(\mathcal{P}_\mathcal{T})}{n^2}\epsilon(\mathcal{P}_\mathcal{T}) + B\left(\frac{n^2 - \hat{p}_n(\mathcal{P}_\mathcal{T})}{n^2} + 2\sqrt{\frac{2K\ln 2 + 2\ln(1/\delta)}{n}}\right).$$

Proof. From the proof of Theorem 7.3, we can easily deduce that:

$$|R^\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}) - R^\ell_{\mathcal{P}_\mathcal{T}}(\mathcal{A}_{\mathcal{P}_\mathcal{T}})| \leq 2B\sum_{i=1}^{K}|\frac{|N_i|}{n} - \mu(C_i)| +$$
$$\left|\sum_{i=1}^{K}\sum_{j=1}^{K}\mathbb{E}_{z,z'\sim P}[\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z')|z \in C_i, z' \in C_j]\frac{|N_i|}{n}\frac{|N_j|}{n} - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_i, z_j)\right|.$$

Then, we have

$$\begin{aligned}
\leq\ & 2B\sum_{i=1}^{K}|\frac{|N_i|}{n} - \mu(C_i)| + \\
& \left|\frac{1}{n^2}\sum_{i=1}^{K}\sum_{j=1}^{K}\sum_{(z_o,z_l)\in\hat{\mathcal{P}}_\mathcal{T}}\sum_{z_o\in N_i}\sum_{z_l\in N_j}\max_{z\in C_i}\max_{z'\in C_j}|\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z') - \ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_o, z_l)|\right| + \\
& \left|\frac{1}{n^2}\sum_{i=1}^{K}\sum_{j=1}^{K}\sum_{(z_o,z_l)\notin\hat{\mathcal{P}}_\mathcal{T}}\sum_{z_o\in N_i}\sum_{z_l\in N_j}\max_{z\in C_i}\max_{z'\in C_j}|\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z') - \ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_o, z_l)|\right| \\
\leq\ & \frac{\hat{p}_n(\mathcal{P}_\mathcal{T})}{n^2}\epsilon(\mathcal{P}_\mathcal{T}) + B\left(\frac{n^2 - \hat{p}_n(\mathcal{P}_\mathcal{T})}{n^2} + 2\sqrt{\frac{2K\ln 2 + 2\ln(1/\delta)}{n}}\right).
\end{aligned}$$

The second inequality is obtained by the triangle inequality, the last one is obtained by the application of Proposition 7.2, the hypothesis of pseudo-robustness and the fact that ℓ is nonnegative and bounded by B and thus $|\ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z, z') - \ell(\mathcal{A}_{\mathcal{P}_\mathcal{T}}, z_o, z_l)| \leq B$. □

B.2.2 Proof of sufficiency of Theorem 7.8

Theorem Given a fixed sequence of training examples $\mathcal{T}^*$, a metric learning method $\mathcal{A}$ generalizes with respect to $\mathcal{P}_{\mathcal{T}^*}$ if and only if it is weakly robust with respect to $\mathcal{P}_{\mathcal{T}^*}$.

Proof. The proof of sufficiency corresponds to the first part of the proof of Theorem 8 of Xu & Mannor (2012). When $\mathcal{A}$ is weakly robust there exists a sequence $\{\mathcal{D}_n\}$ such that

for any $\delta, \epsilon > 0$ there exists $N(\delta, \epsilon)$ such that for all $n > N(\delta, \epsilon)$, $Pr(\mathcal{U}(n) \in \mathcal{D}_n) > 1 - \delta$ and

$$\max_{\hat{\mathcal{T}}(n) \in \mathcal{D}_n} \left| R^{\ell}_{\mathcal{P}_{\hat{\mathcal{T}}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) \right| < \epsilon. \tag{B.7}$$

Therefore for any $n > N(\delta, \epsilon)$,

$$\begin{aligned}
&|R^{\ell}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})| \\
&= |\mathbb{E}_{\mathcal{U}(n)}[R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})] - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})| \\
&= |Pr(\mathcal{U}(n) \notin \mathcal{D}_n)\mathbb{E}[R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{p_{\mathcal{T}^*(n)}})|\mathcal{U}(n) \notin \mathcal{D}_n] \\
&\quad + Pr(\mathcal{U}(n) \in \mathcal{D}_n)\mathbb{E}[R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{p_{\mathcal{T}^*(n)}})|\mathcal{U}(n) \in \mathcal{D}_n] - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{p_{\mathcal{T}^*(n)}})| \\
&\leq Pr(\mathcal{U}(n) \notin \mathcal{D}_n)|\mathbb{E}[R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{p_{\mathcal{T}^*(n)}})|\mathcal{U}(n) \notin \mathcal{D}_n] - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{p_{\mathcal{T}^*(n)}})| + \\
&\quad Pr(\mathcal{U}(n) \in \mathcal{D}_n)|\mathbb{E}[R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{p_{\mathcal{T}^*(n)}})|\mathcal{U}(n) \in \mathcal{D}_n] - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{p_{\mathcal{T}^*(n)}})| \\
&\leq \delta B + \max_{\hat{\mathcal{T}}(n) \in \mathcal{D}_n} |R^{\ell}_{\mathcal{P}_{\hat{\mathcal{T}}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})| \\
&\leq \delta B + \epsilon.
\end{aligned}$$

The first inequality holds because the testing samples $\mathcal{U}(n)$ consist of n instances IID from P. The second equality is obtained by conditional expectation. The next inequality uses the fact that ℓ is nonnegative and upper bounded by B. Finally, we apply (B.7). We thus conclude that $\mathcal{A}$ generalizes for $\mathcal{P}_{\mathcal{T}^*}$ because ϵ and δ can be chosen arbitrarily. □

B.2.3 Proof of Lemma 7.9

Lemma Given $\mathcal{T}^*$, if a learning method is not weakly robust with respect to $\mathcal{P}_{\mathcal{T}^*}$, there exist $\epsilon^*, \delta^* > 0$ such that the following holds for infinitely many n:

$$Pr(|R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})| \geq \epsilon^*) \geq \delta^*.$$

Proof. This proof follows exactly the same principle as the proof of Lemma 2 from Xu & Mannor (2012). By contradiction, assume ϵ^* and δ^* do not exist. Let $\epsilon_v = \delta_v = 1/v$ for $v = 1, 2, ...$, then there exists a non decreasing sequence $\{N(v)\}_{v=1}^{\infty}$ such that for all v, if $n \geq N(v)$ then

$$Pr(|R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})| \geq \epsilon_v) < \delta_v.$$

For each n we define

$$\mathcal{D}^v_n \triangleq \{\hat{\mathcal{T}}(n) | R^{\ell}_{\mathcal{P}_{\hat{\mathcal{T}}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})| < \epsilon_v\}.$$

For each $n \geq N(v)$ we have

$$Pr(\mathcal{U}(n) \in \mathcal{D}^v_n) = 1 - Pr(|R^{\ell}_{\mathcal{P}_{\mathcal{U}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})| \geq \epsilon_v) > 1 - \delta_v.$$

For $n \geq N(1)$, define $\mathcal{D}_n \triangleq \mathcal{D}_n^{v(n)}$, where $v(n) = \max(v|N(v) \leq n; v \leq n)$. Thus for all, $n \geq N(1)$ we have $Pr(\mathcal{U}(n) \in \mathcal{D}_n) > 1 - \delta_{v(n)}$ and

$$\sup_{\hat{\mathcal{T}}(n) \in \mathcal{D}_n} |R^{\ell}_{\mathcal{P}_{\hat{\mathcal{T}}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})| < \epsilon_{v(n)}.$$

Note that $v(n)$ tends to infinity, it follows that $\delta_{v(n)} \to 0$ and $\epsilon_{v(n)} \to 0$. Therefore, $Pr(\mathcal{U}(n) \in \mathcal{D}_n) \to 1$ and

$$\lim_{n \to \infty} \{ \sup_{\hat{\mathcal{T}}(n) \in \mathcal{D}_n} |R^{\ell}_{\mathcal{P}_{\hat{\mathcal{T}}(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}}) - R^{\ell}_{\mathcal{P}_{\mathcal{T}^*(n)}}(\mathcal{A}_{\mathcal{P}_{\mathcal{T}^*(n)}})|\} = 0.$$

That is $\mathcal{A}$ is weakly robust with respect to $\mathcal{P}_{\mathcal{T}}$, which is the desired contradiction. □

B.2.4 Proof of Example 7.2 (L_1 norm)

Example Algorithm (7.7) with $\|\mathbf{M}\| = \|\mathbf{M}\|_1$ is $(|\mathcal{Y}|\mathcal{N}(\gamma, \mathcal{X}, \|\cdot\|_1), \frac{8UR\gamma g_0}{C})$-robust.

Proof. Let $\mathbf{M}^*$ be the solution given training data $\mathcal{P}_{\mathcal{T}}$. Due to optimality of $\mathbf{M}^*$, we have $\|\mathbf{M}^*\|_1 \leq g_0/C$. We can partition $\mathcal{Z}$ as $|\mathcal{Y}|\mathcal{N}(\gamma/2, \mathcal{X}, \|\cdot\|_1)$ sets, such that if z and z' belong to the same set, then $y = y'$ and $\|\mathbf{x} - \mathbf{x}'\|_1 \leq \gamma$. Now, for $z_1, z_2, z_1', z_2' \in \mathcal{Z}$, if $y_1 = y_1'$, $\|\mathbf{x_1} - \mathbf{x_1'}\|_1 \leq \gamma$, $y_2 = y_2'$ and $\|\mathbf{x_2} - \mathbf{x_2'}\|_1 \leq \gamma$, then:

$$\begin{aligned}
&|g(y_1 y_2[1 - d^2_{\mathbf{M}^*}(\mathbf{x_1}, \mathbf{x_2})]) - g(y_1' y_2'[1 - d^2_{\mathbf{M}^*}(\mathbf{x_1'}, \mathbf{x_2'})])| \\
&\leq U(|(\mathbf{x_1} - \mathbf{x_2})^T \mathbf{M}^*(\mathbf{x_1} - \mathbf{x_1'})| + |(\mathbf{x_1} - \mathbf{x_2})^T \mathbf{M}^*(\mathbf{x_2'} - \mathbf{x_2})| \\
&\quad + |(\mathbf{x_1} - \mathbf{x_1'})^T \mathbf{M}^*(\mathbf{x_1'} + \mathbf{x_2'})| + |(\mathbf{x_2'} - \mathbf{x_2})^T \mathbf{M}^*(\mathbf{x_1'} + \mathbf{x_2'})|) \\
&\leq U(\|\mathbf{x_1} - \mathbf{x_2}\|_\infty \|\mathbf{M}^*\|_1 \|\mathbf{x_1} - \mathbf{x_1'}\|_1 + \|\mathbf{x_1} - \mathbf{x_2}\|_\infty \|\mathbf{M}^*\|_1 \|\mathbf{x_2'} - \mathbf{x_2}\|_1 \\
&\quad + \|\mathbf{x_1} - \mathbf{x_1'}\|_1 \|\mathbf{M}^*\|_1 \|\mathbf{x_1'} - \mathbf{x_2'}\|_\infty + \|\mathbf{x_2'} - \mathbf{x_2}\|_1 \|\mathbf{M}^*\|_1 \|\mathbf{x_1'} - \mathbf{x_2'}\|_\infty) \\
&\leq \frac{8UR\gamma g_0}{C}.
\end{aligned}$$

□

B.2.5 Proof of Example 7.3 ($L_{2,1}$ norm and trace norm)

Example Algorithm (7.7) with $\|\mathbf{M}\| = \|\mathbf{M}\|_{2,1}$ or $\|\mathbf{M}\| = \|\mathbf{M}\|_*$ is $(|\mathcal{Y}|\mathcal{N}(\gamma, \mathcal{X}, \|\cdot\|_2), \frac{8UR\gamma g_0}{C})$-robust.

Proof. We can prove the robustness for the $L_{2,1}$ norm and the trace norm in the same way. Let $\|M\|$ be either the $L_{2,1}$ norm or the trace norm and $\mathbf{M}^*$ be the solution given training data $\mathcal{P}_{\mathcal{T}}$. Due to optimality of $\mathbf{M}^*$, we have $\|\mathbf{M}^*\| \leq g_0/C$. We can partition $\mathcal{Z}$ in the same way as in the proof of Example 7.1 and use the inequality

$\|\mathbf{M}^*\|_{\mathcal{F}} \leq \|\mathbf{M}^*\|_{2,1}$ (from Theorem 3 of Feng, 2003) for the $L_{2,1}$ norm or the well-known inequality $\|\mathbf{M}^*\|_{\mathcal{F}} \leq \|\mathbf{M}^*\|_*$ for the trace norm to derive the same bound:

$$\begin{aligned}
&|g(y_1y_2[1-d^2_{\mathbf{M}^*}(\mathbf{x_1},\mathbf{x_2})]) - g(y_1'y_2'[1-d^2_{\mathbf{M}^*}(\mathbf{x_1'},\mathbf{x_2'})])| \\
&\leq\ U(\|\mathbf{x_1}-\mathbf{x_2}\|_2\|\mathbf{M}^*\|_{\mathcal{F}}\|\mathbf{x_1}-\mathbf{x_1'}\|_2 + \|\mathbf{x_1}-\mathbf{x_2}\|_2\|\mathbf{M}^*\|_{\mathcal{F}}\|\mathbf{x_2'}-\mathbf{x_2}\|_2 \\
&\quad + \|\mathbf{x_1}-\mathbf{x_1'}\|_2\|\mathbf{M}^*\|_{\mathcal{F}}\|\mathbf{x_1'}-\mathbf{x_2'}\|_2 + \|\mathbf{x_2'}-\mathbf{x_2}\|_2\|\mathbf{M}^*\|_{\mathcal{F}}\|\mathbf{x_1'}-\mathbf{x_2'}\|_2) \\
&\leq\ U(\|\mathbf{x_1}-\mathbf{x_2}\|_2\|\mathbf{M}^*\|\|\mathbf{x_1}-\mathbf{x_1'}\|_2 + \|\mathbf{x_1}-\mathbf{x_2}\|_2\|\mathbf{M}^*\|\|\mathbf{x_2'}-\mathbf{x_2}\|_2 \\
&\quad + \|\mathbf{x_1}-\mathbf{x_1'}\|_2\|\mathbf{M}^*\|\|\mathbf{x_1'}-\mathbf{x_2'}\|_2 + \|\mathbf{x_2'}-\mathbf{x_2}\|_2\|\mathbf{M}^*\|\|\mathbf{x_1'}-\mathbf{x_2'}\|_2) \\
&\leq\ \frac{8UR\gamma g_0}{C}.
\end{aligned}$$

□

B.2.6 Proof of Example 7.4 (Kernelization)

Example Consider the kernelized version of Algorithm (7.7):

$$\min_{\mathbf{M}\succeq 0}\ \frac{1}{n^2}\sum_{(z_i,z_j)\in\mathcal{P}_{\mathcal{T}}} g(y_iy_j[1-d^2_{\mathbf{M}}(\phi(\mathbf{x_i}),\phi(\mathbf{x_j}))]) \quad + \quad C\|\mathbf{M}\|_{\mathbb{H}},$$

where $\phi(\cdot)$ is a feature mapping to a kernel space $\mathbb{H}$, $\|\cdot\|_{\mathbb{H}}$ the norm function of $\mathbb{H}$ and $k(\cdot,\cdot)$ the kernel function. Consider a cover of $\mathcal{X}$ by $\|\cdot\|_2$ ($\mathcal{X}$ being compact) and let

$$f_{\mathbb{H}}(\gamma) = \max_{\mathbf{a},\mathbf{b}\in\mathcal{X},\|\mathbf{a}-\mathbf{b}\|_2\leq\gamma} K(\mathbf{a},\mathbf{a}) + K(\mathbf{b},\mathbf{b}) - 2K(\mathbf{a},\mathbf{b}) \quad \text{and} \quad B_\gamma = \max_{x\in\mathcal{X}}\sqrt{K(\mathbf{x},\mathbf{x})}.$$

If the kernel function is continuous, B_γ and $f_{\mathbb{H}}$ are finite for any $\gamma > 0$ and thus the algorithm is $(|\mathcal{Y}|\mathcal{N}(\gamma,\mathcal{X},\|\cdot\|_2), \frac{8UB_\gamma\sqrt{f_{\mathbb{H}}}g_0}{C})$-robust.

Proof. We assume $\mathbb{H}$ to be an Hilbert space with an inner product operator $\langle\cdot,\cdot\rangle$. The mapping ϕ is continuous from $\mathcal{X}$ to $\mathbb{H}$. The norm $\|\cdot\|_{\mathbb{H}} : \mathbb{H} \to \mathbb{R}$ is defined as $\|\mathbf{x}\|_{\mathbb{H}} = \sqrt{\langle\mathbf{x},\mathbf{x}\rangle}$ for all $\mathbf{x} \in \mathbb{H}$, for matrices $\|\mathbf{M}\|_{\mathbb{H}}$ we take the Frobenius norm. The kernel function is defined as $K(\mathbf{x_1},\mathbf{x_2}) = \langle\phi(\mathbf{x_1}),\phi(\mathbf{x_2})\rangle$.

B_γ and $f_{\mathbb{H}}(\gamma)$ are finite by the compactness of $\mathcal{X}$ and continuity of $K(\cdot,\cdot)$. Let $\mathbf{M}^*$ be the solution given training data $\mathcal{P}_{\mathcal{T}}$, by the optimality of $\mathbf{M}^*$ and using the same trick as for the previous example proofs we have $\|\mathbf{M}^*\|_{\mathbb{H}} \leq g_0/c$. Then, by considering a partition of $\mathcal{Z}$ into $|\mathcal{Y}|\mathcal{N}(\gamma/2,\mathcal{X},\|\cdot\|_2)$ disjoint subsets such that if $(\mathbf{x_1},y_1)$ and $(\mathbf{x_2},y_2)$ belong to the same set then $y_1 = y_2$ and $\|\mathbf{x_1}-\mathbf{x_2}\|_2 \leq \gamma$.

We have:

$$
\begin{aligned}
&|g(y_1y_2[1-d^2_{\mathbf{M}^*}(\phi(\mathbf{x_1}),\phi(\mathbf{x_2}))])-g(y'_1y'_2[1-d^2_{\mathbf{M}^*}(\phi(\mathbf{x'_1}),\phi(\mathbf{x'_2}))])| \\
&\leq\ U(|(\phi(\mathbf{x_1})-\phi(\mathbf{x_2}))^T\mathbf{M}^*(\phi(\mathbf{x_1})-\phi(\mathbf{x'_1}))|+|(\phi(\mathbf{x_1})-\phi(\mathbf{x_2}))^T\mathbf{M}^*(\phi(\mathbf{x'_2})-\phi(\mathbf{x_2}))| \\
&\quad +|(\phi(\mathbf{x_1})-\phi(\mathbf{x'_1}))^T\mathbf{M}^*(\phi(\mathbf{x'_1})+\phi(\mathbf{x'_2}))|+|(\phi(\mathbf{x'_2})-\phi(\mathbf{x_2}))^T\mathbf{M}^*(\phi(\mathbf{x'_1})+\phi(\mathbf{x'_2}))|) \\
&\leq\ U(|\phi(\mathbf{x_1})^T\mathbf{M}^*(\phi(\mathbf{x_1})-\phi(\mathbf{x'_1}))|+|\phi(\mathbf{x_2})^T\mathbf{M}^*(\phi(\mathbf{x_1})-\phi(\mathbf{x'_1}))|+ \qquad \text{(B.8)} \\
&\quad |\phi(\mathbf{x_1})^T\mathbf{M}^*(\phi(\mathbf{x'_2})\phi(\mathbf{x_2}))|+|\phi(\mathbf{x_2})^T\mathbf{M}^*(\phi(\mathbf{x'_2})-\phi(\mathbf{x_2}))|+ \\
&\quad |(\phi(\mathbf{x_1})-\phi(\mathbf{x'_1}))^T\mathbf{M}^*\phi(\mathbf{x'_1})|+|(\phi(\mathbf{x_1})-\phi(\mathbf{x'_1}))^T\mathbf{M}^*\phi(\mathbf{x'_2})|+ \\
&\quad |(\phi(\mathbf{x'_2})-\phi(\mathbf{x_2}))^T\mathbf{M}^*\phi(\mathbf{x'_1})|+|(\phi(\mathbf{x'_2})-\phi(\mathbf{x_2}))^T\mathbf{M}^*\phi(\mathbf{x'_2})|).
\end{aligned}
$$

Then, note that

$$
\begin{aligned}
|\phi(\mathbf{x_1})^T\mathbf{M}^*(\phi(\mathbf{x_1})-\phi(\mathbf{x'_1}))| &\leq \sqrt{\langle\phi(\mathbf{x_1}),\phi(\mathbf{x_1})\rangle}\|\mathbf{M}^*\|_{\mathbb{H}}\sqrt{\langle\phi(\mathbf{x'_1})-\phi(\mathbf{x'_2}),\phi(\mathbf{x'_1})-\phi(\mathbf{x'_2})\rangle} \\
&\leq B_\gamma\frac{g_o}{C}\sqrt{f_{\mathbb{H}}(\gamma)}.
\end{aligned}
$$

Thus, by applying the same principle to all the terms in the right part of inequality (B.8), we obtain:

$$
|g(y_1y_2[1-d^2_{\mathbf{M}^*}(\phi(\mathbf{x_1}),\phi(\mathbf{x_2}))])-g(y_{ij}[1-d^2_{\mathbf{M}^*}(\phi(\mathbf{x'_1}),\phi(\mathbf{x'_2}))])| \leq \frac{8UB_\gamma\sqrt{f_{\mathbb{H}}(\gamma)}g_0}{C}.
$$

□

Bibliography

Noga Alon, Shai Ben-David, Nicol Cesa-Bianchi, and David Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM*, 44(4):615–631, 1997.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

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, 2009.

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, 2006.

Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. A Theory of Learning with Similarity Functions. *Machine Learning Journal (MLJ)*, 72:89–112, 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, 2008b.

Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. A Discriminative Framework for Clustering via Similarity Functions. In *ACM Symposium on Theory of Computing (STOC)*, pages 671–680, 2008c.

Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with Bregman Divergences. *Journal of Machine Learning Research (JMLR)*, 6:1705–1749, 2005.

Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research (JMLR)*, 3: 463–482, 2002.

Aurélien Bellet, Marc Bernard, Thierry Murgue, and Marc Sebban. Apprentissage de noyaux d'édition de séquences. In *French Conference on Machine Learning (CAp)*, pages 93–108, 2009. Best paper award.

Aurélien Bellet, Marc Bernard, Thierry Murgue, and Marc Sebban. Learning state machine-based string edit kernels. *Pattern Recognition (PR)*, 43(6):2330–2339, 2010.

Aurélien Bellet and Amaury Habrard. Robustness and Generalization for Metric Learning. Technical report, University of Saint-Etienne, September 2012. arXiv:1209.1086.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. An Experimental Study on Learning with Good Edit Similarity Functions. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 126–133, 2011a.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. Apprentissage Parcimonieux à partir de Fonctions de Similarité d'Édition (ϵ, γ, τ)-Good. In *French Conference on Machine Learning (CAp)*, pages 103–118, 2011b.

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, 2011c.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. Apprentissage de bonnes similarités pour la classification linéaire parcimonieuse. In *French Conference on Machine Learning (CAp)*, pages 302–317, 2012a.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. Good edit similarity learning by loss minimization. *Machine Learning Journal (MLJ)*, 89(1):5–35, 2012b.

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)*, 2012c.

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

Shai Ben-David, Nadav Eiron, and Philip M. Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences (JCSS)*, 66(3): 496–514, 2003.

Shai Ben-David, David Loker, Nathan Srebro, and Karthik Sridharan. Minimizing The Misclassification Error Rate Using a Surrogate Convex Loss. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

Marc Bernard, Laurent Boyer, Amaury Habrard, and Marc Sebban. Learning probabilistic models of tree edit distance. *Pattern Recognition (PR)*, 41(8):2611–2629, 2008.

Marc Bernard, Amaury Habrard, and Marc Sebban. Learning Stochastic Tree Edit Distance. In *Proceedings of the 17th European Conference on Machine Learning (ECML)*, pages 42–53, 2006.

Enrico Bertini, Andrada Tatu, and Daniel Keim. Quality Metrics in High-Dimensional Data Visualization: An Overview and Systematization. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 17(12):2203–2212, 2011.

Wei Bian. Constrained Empirical Risk Minimization Framework for Distance Metric Learning. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 23(8):1194–1205, 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, 2011.

Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative Learning Under Covariate Shift. *Journal of Machine Learning Research (JMLR)*, 10:2137–2155, 2009.

Mikhail Bilenko and Raymond J. Mooney. Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 39–48, 2003.

Philip Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science (TCS)*, 337(1-3):217–239, 2005.

Stéphane Boucheron, Gábor Lugosi, and Olivier Bousquet. Concentration Inequalities. In *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Computer Science*, pages 208–240, 2004.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to Statistical Learning Theory. In *Advanced Lectures on Machine Learning*, volume 3176, pages 169–207, 2003.

Olivier Bousquet and André Elisseeff. Algorithmic Stability and Generalization Performance. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 196–202, 2001.

Olivier Bousquet and André Elisseeff. Stability and Generalization. *Journal of Machine Learning Research (JMLR)*, 2:499–526, 2002.

Laurent Boyer, Yann Esposito, Amaury Habrard, José Oncina, and Marc Sebban. SEDiL: Software for Edit Distance Learning. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 672–677, 2008.

Laurent Boyer, Amaury Habrard, and Marc Sebban. Learning Metrics between Tree Structured Data: Application to Image Recognition. In *Proceedings of the 18th European Conference on Machine Learning (ECML)*, pages 54–66, 2007.

Lev M. Bregman. The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

Bin Cao, Xiaochuan Ni, Jian-Tao Sun, Gang Wang, and Qiang Yang. Distance Metric Learning under Covariate Shift. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1204–1210, 2011.

Qiong Cao, Yiming Ying, and Peng Li. Distance Metric Learning Revisited. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 283–298, 2012.

Nicol Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the Generalization Ability of On-Line Learning Algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 359–366, 2001.

Nicol Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the Generalization Ability of On-Line Learning Algorithms. *IEEE Transactions on Information Theory (TIT)*, 50(9):2050–2057, 2004.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006.

Ratthachat Chatpatanasiri, Teesid Korsrilabutr, Pasakorn Tangchanachaianan, and Boonserm Kijsirikul. A new kernelization framework for Mahalanobis distance learning algorithms. *Neurocomputing*, 73:1570–1579, 2010.

Gal Chechik, Uri Shalit, Varun Sharma, and Samy Bengio. An Online Algorithm for Large Scale Image Similarity Learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22, pages 306–314, 2009.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large Scale Online Learning of Image Similarity Through Ranking. *Journal of Machine Learning Research (JMLR)*, 11:1109–1135, 2010.

Jianhui Chen and Jieping Ye. Training SVM with indefinite kernels. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 136–143, 2008.

Yihua Chen, Maya R. Gupta, and Benjamin Recht. Learning kernels from indefinite similarities. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 145–152, 2009.

Michael Collins and Nigel Duffy. Convolution Kernels for Natural Language. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 625–632, 2001.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational Kernels: Theory and Algorithms. *Journal of Machine Learning Research (JMLR)*, 5:1035–1062, 2004.

Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning Journal (MLJ)*, 20(3):273–297, 1995.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory (TIT)*, 13(1):21–27, 1967.

Koby Crammer and Gal Chechik. Adaptive Regularization for Weight Matrices. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

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.

Bo Dai, Makoto Yamada, Gang Niu, and Masashi Sugiyama. Information-theoretic Semi-supervised Metric Learning via Entropy Regularization. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

Nilesh N. Dalvi, Philip Bohannon, and Fei Sha. Robust web extraction: an approach based on a probabilistic tree-edit model. In *Proceedings of the ACM SIGMOD International Conference on Management of data (COMAD)*, pages 335–348, 2009.

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, 2007.

Margaret O. Dayhoff, Robert M. Schwartz, and Bruce C. Orcutt. A model of evolutionary change in proteins. *Atlas of protein sequence and structure*, 5(3):345–351, 1978.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

Jia Deng, Alexander C. Berg, and Li Fei-Fei. Hierarchical semantic indexing for large scale image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 785–792, 2011.

François Denis, Yann Esposito, and Amaury Habrard. Learning Rational Stochastic Languages. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT)*, pages 274–288, 2006.

François Denis, Edouard Gilbert, Amaury Habrard, Faissal Ouardi, and Marc Tommasi. Relevant Representations for the Inference of Rational Stochastic Tree Languages. In *Proceedings of the 9th International Colloquium on Grammatical Inference (ICGI)*, pages 57–70, 2008.

Huyen Do, Alexandros Kalousis, Jun Wang, and Adam Woznica. A metric learning perspective of SVM: on the relation of LMNN and SVM. *Journal of Machine Learning Research (JMLR)*, 22:308–317, 2012.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari. Composite Objective Mirror Descent. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*, pages 14–26, 2010.

Charles Elkan. Using the Triangle Inequality to Accelerate k-Means. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 147–153, 2003.

Martin Emms. On Stochastic Tree Distances and Their Training via Expectation-Maximisation. In *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, pages 144–153, 2012.

Martin Emms and Hector-Hugo Franco-Penya. On Order Equivalences between Distance and Similarity Measures on Sequences and Trees. In *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, pages 15–24, 2012.

Kousha Etessami and Mihalis Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1):1, 2009.

Bao Q. Feng. Equivalence constants for certain matrix norms. *Linear Algebra and Its Applications*, 374:247–253, 2003.

Aurélie Fischer. Quantization and clustering with Bregman divergences. *Journal of Multivariate Analysis (JMVA)*, 101(9):2207–2221, 2010.

Herbert Freeman. Computer Processing of Line-Drawing Images. *ACM Computing Surveys*, 6:57–97, 1974.

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, 1995.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics*, 38(2):337–407, 2000.

Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning Globally-Consistent Local Distance Functions for Shape-Based Image Retrieval and Classification. In *Proceedings of the 1th IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and Applications (PAA)*, 13(1):113–129, 2010.

Bo Geng, Dacheng Tao, and Chao Xu. DAML: Domain Adaptation Metric Learning. *IEEE Transactions on Image Processing (TIP)*, 20(10):2980–2989, 2011.

Zoubin Ghahramani. Unsupervised Learning. In *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Computer Science*, pages 72–112, 2003.

Amir Globerson and Sam T. Roweis. Metric Learning by Collapsing Classes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 18, pages 451–458, 2005.

Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood Components Analysis. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 513–520, 2004.

Mehmet Gönen and Ethem Alpaydn. Multiple Kernel Learning Algorithms. *Journal of Machine Learning Research (JMLR)*, 12:2211–2268, 2011.

Yves Grandvalet. Sparsity in learning. Statlearn'11 Workshop on Challenging problems in Statistical Learning, 2011.

Matthieu Guillaumin, Jakob J. Verbeek, and Cordelia Schmid. Is that you? Metric learning approaches for face identification. In *Proceddings of the 11th International Conference on Computer Vision (ICCV)*, pages 498–505, 2009.

David Haussler. Convolution Kernels on Discrete Structure. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, July 1999.

Steven Henikoff and Jorja G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89(22):10915–10919, 1992.

Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems (NIPS)*, volume 11, 1998.

Prateek Jain, Brian Kulis, Inderjit S. Dhillon, and Kristen Grauman. Online Metric Learning and Fast Similarity Search. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, pages 761–768, 2008.

Rong Jin, Shijun Wang, and Yang Zhou. Regularized Distance Metric Learning: Theory and Algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22, pages 862–870, 2009.

Purushottam Kar and Prateek Jain. Similarity-based Learning via Data Driven Embeddings. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, 2011.

Purushottam Kar and Prateek Jain. Supervised Learning with Similarity Functions. In *Advances in Neural Information Processing Systems (NIPS)*, volume 25, pages 215–223, 2012.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized Kernels Between Labeled Graphs. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 321–328, 2003.

Andrei N. Kolmogorov and Vassili M. Tikhomirov. ϵ-entropy and ϵ-capacity of sets in functional spaces. *American Mathematical Society Translations*, 2(17):277–364, 1961.

Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory (TIT)*, 47(5):1902–1914, 2001.

Brian Kulis, Kate Saenko, and Trevor Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1785–1792, 2011.

Brian Kulis, Mátyás A. Sustik, and Inderjit S. Dhillon. Low-Rank Kernel Learning with Bregman Matrix Divergences. *Journal of Machine Learning Research (JMLR)*, 10: 341–376, 2009.

Brian Kulis, Mtys A. Sustik, and Inderjit S. Dhillon. Learning low-rank kernel matrices. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 505–512, 2006.

Gautam Kunapuli and Jude Shavlik. Mirror Descent for Metric Learning: A Unified Approach. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Database (ECML/PKDD)*, pages 859–874, 2012.

Jim Z. C. Lai, Yi-Ching Liaw, and Julie Liu. Fast k-nearest-neighbor search based on projection and triangular inequality. *Pattern Recognition (PR)*, 40(2):351–359, 2007.

Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the Kernel Matrix with Semidefinite Programming. *Journal of Machine Learning Research (JMLR)*, 5:27–72, 2004.

Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the Kernel Matrix with Semi-Definite Programming. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 323–330, 2002.

John Langford. Tutorial on Practical Prediction Theory for Classification. *Journal of Machine Learning Research (JMLR)*, 6:273–306, 2005.

Christina S. Leslie, Eleazar Eskin, and William S. Noble. The Spectrum Kernel: A String Kernel for SVM Protein Classification. In *Pacific Symposium on Biocomputing*, pages 566–575, 2002a.

Christina S. Leslie, Eleazar Eskin, Jason Weston, and William S. Noble. Mismatch String Kernels for SVM Protein Classification. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 1417–1424, 2002b.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Doklandy*, 6:707–710, 1966.

Haifeng Li and Tao Jiang. A class of edit kernels for SVMs to predict translation initiation sites in eukaryotic mRNAs. In *Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 262–271, 2004.

Xi Li, Chunhua Shen, Qinfeng Shi, Anthony Dick, and Anton van den Hengel. Non-sparse Linear Representations for Visual Tracking with Online Reservoir Metric Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1760–1767, 2012.

Nick Littlestone. Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. *Machine Learning Journal (MLJ)*, 2(4):285–318, 1988.

Wei Liu, Shiqian Ma, Dacheng Tao, Jianzhuang Liu, and Peng Liu. Semi-Supervised Sparse Metric Learning using Alternating Linearization Optimization. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1139–1148, 2010.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory (TIT)*, 28:129–137, 1982.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text Classification using String Kernels. *Journal of Machine Learning Research (JMLR)*, 2:419–444, 2002.

Ronny Luss and Alexandre d'Aspremont. Support Vector Machine Classification with Indefinite Kernels. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, 2007.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain Adaptation: Learning Bounds and Algorithms. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT)*, 2009.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. A Conditional Random Field for Discriminatively-trained Finite-state String Edit Distance. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 388–395, 2005.

Colin McDiarmid. *Surveys in Combinatorics*, chapter On the method of bounded differences, pages 148–188. Cambridge University Press, 1989.

Luisa Micó and Jose Oncina. Comparison of fast nearest neighbour classifiers for handwritten character recognition. *Pattern Recognition Letters (PRL)*, 19:351–356, 1998.

Luisa Micó, Jose Oncina, and Enrique Vidal. A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements. *Pattern Recognition Letters (PRL)*, 15(1):9–17, 1994.

Mehryar Mohri and Afshin Rostamizadeh. Stability bounds for non-i.i.d. processes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, 2007.

Mehryar Mohri and Afshin Rostamizadeh. Stability Bounds for Stationary ϕ-mixing and β-mixing Processes. *Journal of Machine Learning Research (JMLR)*, 11:789–814, 2010.

David W. Mount. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, 2nd edition, 2004.

Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology (JMB)*, 48(3):443–453, 1970.

Michel Neuhaus and Horst Bunke. Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition (PR)*, 39:1852–1863, 2006.

Michel Neuhaus and Horst Bunke. Automatic learning of cost functions for graph edit distance. *Journal of Information Science (JIS)*, 177(1):239–247, 2007.

Frank Nielsen and Richard Nock. Sided and symmetrized Bregman centroids. *IEEE Transactions on Information Theory (TIT)*, 55(6):2882–2904, 2009.

Jose Oncina and Marc Sebban. Learning Stochastic Edit Distance: application in handwritten character recognition. *Pattern Recognition (PR)*, 39(9):1575–1587, 2006.

Cheng Soon Ong, Xavier Mary, Stphane Canu, and Alexander J. Smola. Learning with non-positive kernels. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Hyperkernels. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 478–485, 2002.

Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Learning the Kernel with Hyperkernels. *Journal of Machine Learning Research (JMLR)*, 6:1043–1071, 2005.

Sinno J. Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(10):1345–1359, 2010.

Shibin Parameswaran and Kilian Q. Weinberger. Large Margin Multi-Task Metric Learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 23, pages 1867–1875, 2010.

Kyoungup Park, Chunhua Shen, Zhihui Hao, and Junae Kim. Efficiently Learning a Distance Metric for Large Margin Nearest Neighbor Classification. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 2011.

Mateusz Pawlik and Nikolaus Augsten. RTED: a robust algorithm for the tree edit distance. *Proceedings of the VLDB Endowment*, 5(4):334–345, 2011.

Karl Pearson. On Lines and Planes of Closest Fit to Points in Space. *Philosophical Magazine*, 2(6):559–572, 1901.

Ali M. Qamar and Eric Gaussier. Online and Batch Learning of Generalized Cosine Similarities. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 926–931, 2009.

Ali M. Qamar and Eric Gaussier. RELIEF Algorithm and Similarity Learning for k-NN. *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)*, 4:445–458, 2012.

Ali M. Qamar, Eric Gaussier, Jean-Pierre Chevallet, and Joo-Hwee Lim. Similarity Learning for Nearest Neighbor Classification. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 983–988, 2008.

Guo-Jun Qi, Jinhui Tang, Zheng-Jun Zha, Tat-Seng Chua, and Hong-Jiang Zhang. An Efficient Sparse Metric Learning in High-Dimensional Space via l1-Penalized Log-Determinant Regularization. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009.

Eric S. Ristad and Peter N. Yianilos. Learning String-Edit Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(5):522–532, 1998.

Romer Rosales and Glenn Fung. Learning Sparse Metrics via Linear Programming. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 367–373, 2006.

Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are Loss Functions All the Same? *Neural Computation (NECO)*, 16(5):1063–1076, 2004.

Hiroto Saigo, Jean-Philippe Vert, and Tatsuya Akutsu. Optimizing amino acid substitution matrices with a local alignment kernel. *Bioinformatics*, 7(246):1–12, 2006.

Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

Gerard Salton, Andrew Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. MIT Press, 2012.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation (NECO)*, 10(1):1299–1319, 1998.

Bernhard Schölkopf and Alexander J. Smola. *Learning With Kernels, Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

Bernhard Schölkopf, Jason Weston, Eleazar Eskin, Christina Leslie, and William S. Noble. A Kernel Approach for Learning from almost Orthogonal Patterns. In *Proceedings of the 13th European Conference on Machine Learning (ECML)*, pages 511–528, 2002.

Matthew Schultz and Thorsten Joachims. Learning a Distance Metric from Relative Comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, 2003.

Stanley M. Selkow. The tree-to-tree editing problem. *Information Processing Letters*, 6 (6):184–186, 1977.

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)*, 2004.

Uri Shalit, Daphna Weinshall, and Gal Chechik. Online Learning in The Manifold of Low-Rank Matrices. In *Advances in Neural Information Processing Systems (NIPS)*, volume 23, pages 2128–2136, 2010.

Uri Shalit, Daphna Weinshall, and Gal Chechik. Online Learning in the Embedded Manifold of Low-rank Matrices. *Journal of Machine Learning Research (JMLR)*, 13: 429–458, 2012.

Chunhua Shen, Junae Kim, Lei Wang, and Anton van den Hengel. Positive Semidefinite Metric Learning with Boosting. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22, pages 1651–1660, 2009.

Chunhua Shen, Junae Kim, Lei Wang, and Anton van den Hengel. Positive Semidefinite Metric Learning Using Boosting-like Algorithms. *Journal of Machine Learning Research (JMLR)*, 13:1007–1036, 2012.

Kilho Shin, Marco Cuturi, and Tetsuji Kuboyama. Mapping kernels for trees. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 961–968, 2011.

Kilho Shin and Tetsuji Kuboyama. A generalization of Haussler's convolution kernel: mapping kernel. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 944–951, 2008.

Josef Sivic and Andrew Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31:591–606, 2009.

Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology (JMB)*, 147(1):195–197, 1981.

Ingo Steinwart. Sparseness of Support Vector Machines. *Journal of Machine Learning Research (JMLR)*, 4:1071–1105, 2003.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

Atsuhiro Takasu. Bayesian Similarity Model Estimation for Approximate Recognized Text Search. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR)*, pages 611–615, 2009.

Koji Tsuda, Taishin Kin, and Kiyoshi Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18(1):268–275, 2002.

Koji Tsuda, Gunnar Rätsch, and Manfred K. Warmuth. Matrix Exponentiated Gradient Updates for On-line Learning and Bregman Projection. *Journal of Machine Learning Research (JMLR)*, 6:995–1018, 2005.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

Aad W. van der Vaart and Jon A. Wellner. *Weak convergence and empirical processes.* Springer, 2000.

Lieven Vandenberghe and Stephen Boyd. Semidefinite Programming. *SIAM Review (SIREV)*, 38(1):49–95, 1996.

Vladimir N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, 1982.

Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1998.

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.

Jarkko Venna, Jaakko Peltonen, Kristian Nybo, Helena Aidos, and Samuel Kaski. Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization. *Journal of Machine Learning Research (JMLR)*, 11:451–490, 2010.

Nakul Verma, Dhruv Mahajan, Sundararajan Sellamanickam, and Vinod Nair. Learning Hierarchical Similarity Metrics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2280–2287, 2012.

Jun Wang, Huyen T. Do, Adam Woznica, and Alexandros Kalousis. Metric Learning with Multiple Kernels. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, pages 1170–1178, 2011.

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, 2012.

Liwei Wang, Masashi Sugiyama, Cheng Yang, Kohei Hatano, and Jufu Feng. Theory and Algorithm for Learning with Dissimilarity Functions. *Neural Computation (NECO)*, 21(5):1459–1484, 2009.

Liwei Wang, Cheng Yang, and Jufu Feng. On Learning with Dissimilarity Functions. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 991–998, 2007.

Xueyi Wang. Fast Exact k-Nearest Neighbors Algorithm for High Dimensional Search Using k-Means Clustering and Triangle Inequality. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, pages 1293–1299, 2011.

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)*, volume 18, pages 1473–1480, 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, 2008.

Kilian Q. Weinberger and Lawrence K. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research (JMLR)*, 10: 207–244, 2009.

Lei Wu, Steven C.-H. Hoi, Rong Jin, Jianke Zhu, and Nenghai Yu. Learning Bregman Distance Functions for Semi-Supervised Clustering. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 24(3):478–491, 2012.

Lei Wu, Rong Jin, Steven C.-H. Hoi, Jianke Zhu, and Nenghai Yu. Learning Bregman Distance Functions and Its Application for Semi-Supervised Clustering. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22, pages 2089–2097, 2009.

Lin Xiao. Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization. *Journal of Machine Learning Research (JMLR)*, 11:2543–2596, 2010.

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart J. 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, Constantine Caramanis, and Shie Mannor. Sparse Algorithms Are Not Stable: A No-Free-Lunch Theorem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(1):187–193, 2012a.

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.

Zhixiang Xu, Kilian Q. Weinberger, and Olivier Chapelle. Distance Metric Learning for Kernel Machines. arXiv:1208.3422, 2012b.

Haiqin Yang, Zenglin Xu, Irwin King, and Michael R. Lyu. Online Learning for Group Lasso. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 1191–1198, 2010.

Peipei Yang, Kaizhu Huang, and Cheng-Lin Liu. Geometry Preserving Multi-task Metric Learning. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 648–664, 2012.

Rui Yang, Panos Kalnis, and Anthony K. H. Tung. Similarity evaluation on tree-structured data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (COMAD)*, pages 754–765, 2005.

Yiming Ying, Kaizhu Huang, and Colin Campbell. Sparse Metric Learning via Smooth Optimization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22, pages 2214–2222, 2009.

Yiming Ying and Peng Li. Distance Metric Learning with Eigenvalue Optimization. *Journal of Machine Learning Research (JMLR)*, 13:1–26, 2012.

Zheng-Jun Zha, Tao Mei, Meng Wang, Zengfu Wang, and Xian-Sheng Hua. Robust Distance Metric Learning with Auxiliary Knowledge. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1327–1332, 2009.

Changshui Zhang, Feiping Nie, and Shiming Xiang. A general kernelization framework for learning algorithms based on kernel PCA. *Neurocomputing*, 73(4–6):959–967, 2010.

Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing (SICOMP)*, 18(6): 1245–1262, 1989.

Ji Zhu, Saharon Rosset, Trevor Hastie, and Robert Tibshirani. 1-norm Support Vector Machines. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, pages 49–56, 2003.

Abstract In recent years, the crucial importance of metrics in machine learning algorithms has led to an increasing interest in optimizing distance and similarity functions using knowledge from training data to make them suitable for the problem at hand. This area of research is known as *metric learning*. Existing methods typically aim at optimizing the parameters of a given metric with respect to some local constraints over the training sample. The learned metrics are generally used in nearest-neighbor and clustering algorithms. When data consist of feature vectors, a large body of work has focused on learning a Mahalanobis distance, which is parameterized by a positive semi-definite matrix. Recent methods offer good scalability to large datasets. Less work has been devoted to metric learning from structured objects (such as strings or trees), because it often involves complex procedures. Most of the work has focused on optimizing a notion of edit distance, which measures (in terms of number of operations) the cost of turning an object into another. We identify two important limitations of current supervised metric learning approaches. First, they allow to improve the performance of *local* algorithms such as k-nearest neighbors, but metric learning for *global* algorithms (such as linear classifiers) has not really been studied so far. Second, and perhaps more importantly, the question of the generalization ability of metric learning methods has been largely ignored. In this thesis, we propose theoretical and algorithmic contributions that address these limitations. Our first contribution is the derivation of a new kernel function built from learned edit probabilities. Unlike other string kernels, it is guaranteed to be valid and parameter-free. Our second contribution is a novel framework for learning string and tree edit similarities inspired by the recent theory of (ϵ, γ, τ)-good similarity functions and formulated as a convex optimization problem. Using uniform stability arguments, we establish theoretical guarantees for the learned similarity that give a bound on the generalization error of a linear classifier built from that similarity. In our third contribution, we extend the same ideas to metric learning from feature vectors by proposing a bilinear similarity learning method that efficiently optimizes the (ϵ, γ, τ)-goodness. The similarity is learned based on global constraints that are more appropriate to linear classification. Generalization guarantees are derived for our approach, highlighting that our method minimizes a tighter bound on the generalization error of the classifier. Our last contribution is a framework for establishing generalization bounds for a large class of existing metric learning algorithms. It is based on a simple adaptation of the notion of algorithmic robustness and allows the derivation of bounds for various loss functions and regularizers.

Résumé Ces dernières années, l'importance cruciale des métriques en apprentissage automatique a mené à un intérêt grandissant pour l'optimisation de distances et de similarités en utilisant l'information contenue dans des données d'apprentissage pour les rendre adaptées au problème traité. Ce domaine de recherche est souvent appelé *apprentissage de métriques*. En général, les méthodes existantes optimisent les paramètres d'une métrique devant respecter des contraintes locales sur les données d'apprentissage. Les métriques ainsi apprises sont généralement utilisées dans des algorithms de plus proches voisins ou de clustering. Concernant les données numériques, beaucoup de travaux ont porté sur l'apprentissage de distance de Mahalanobis, paramétrisée par une matrice positive semi-définie. Les méthodes récentes sont capables de traiter des jeux de données de grande taille. Moins de travaux ont été dédiés à l'apprentissage de métriques pour les données structurées (comme les chaînes ou les arbres), car cela implique souvent des procédures plus complexes. La plupart des travaux portent sur l'optimisation d'une notion de distance d'édition, qui mesure (en termes de nombre d'opérations) le coût de transformer un objet en un autre. Au regard de l'état de l'art, nous avons identifié deux limites importantes des approches actuelles. Premièrement, elles permettent d'améliorer la performance d'algorithmes *locaux* comme les k plus proches voisins, mais l'apprentissage de métriques pour des algorithmes *globaux* (comme les classifieurs linéaires) n'a pour l'instant pas été beaucoup étudié. Le deuxième point, sans doute le plus important, est que la question de la capacité de généralisation des méthodes d'apprentissage de métriques a été largement ignorée. Dans cette thèse, nous proposons des contributions théoriques et algorithmiques qui répondent à ces limites. Notre première contribution est la construction d'un nouveau noyau construit à partir de probabilités d'édition apprises. A l'inverse d'autres noyaux entre chaînes, sa validité est garantie et il ne comporte aucun paramètre. Notre deuxième contribution est une nouvelle approche d'apprentissage de similarités d'édition pour les chaînes et les arbres inspirée par la théorie des (ϵ, γ, τ)-bonnes fonctions de similarité et formulée comme un problème d'optimisation convexe. En utilisant la notion de stabilité uniforme, nous établissons des garanties théoriques pour la similarité apprise qui donne une borne sur l'erreur en généralisation d'un classifieur linéaire construit à partir de cette similarité. Dans notre troisième contribution, nous étendons ces principes à l'apprentissage de métriques pour les données numériques en proposant une méthode d'apprentissage de similarité bilinéaire qui optimise efficacement l'(ϵ, γ, τ)-goodness. La similarité est apprise sous contraintes globales, plus appropriées à la classification linéaire. Nous dérivons des garanties théoriques pour notre approche, qui donnent de meilleurs bornes en généralisation pour le classifieur que dans le cas des données structurées. Notre dernière contribution est un cadre théorique permettant d'établir des bornes en généralisation pour de nombreuses méthodes existantes d'apprentissage de métriques. Ce cadre est basé sur la notion de robustesse algorithmique et permet la dérivation de bornes pour des fonctions de perte et des régulariseurs variés.