École Doctorale ED488 Sciences, Ingénierie, Santé

Numéro d'ordre NNT : 2020LYSES044

# Learning Tailored Data Representations from Few Labeled Examples

# Construction de Représentations de Données Adaptées dans le Cadre de peu d'Exemples Étiquetés

Thèse préparée par **Léo Gautheron**
au sein de l'**Université Jean Monnet de Saint-Étienne**
pour obtenir le grade de :

**Docteur de l'Université de Lyon**
Spécialité : **Informatique**

Univ Lyon, UJM-Saint-Etienne, CNRS, Institut d'Optique Graduate School,
Laboratoire Hubert Curien UMR 5516, F-42023, SAINT-ETIENNE, France.

Thèse soutenue publiquement le 8 décembre 2020 devant le jury composé de :

| | | |
|---|---|---|
| Paulo GONÇALVES | Directeur de recherche, INRIA Rhône-Alpes | Rapporteur |
| Amaury HABRARD | Professeur, Université de Saint-Étienne | Co-Directeur |
| Emilie MORVANT | Maître de conférences, Université de Saint-Étienne | Co-Encadrante |
| Marc SEBBAN | Professeur, Université de Saint-Étienne | Directeur |
| Christine SOLNON | Professeur, INSA de Lyon | Examinatrice |
| Marc TOMMASI | Professeur, Université de Lille | Rapporteur |

# Table of Contents

# Introduction

Machine learning consists in the study and design of algorithms that build models able to handle non trivial tasks as well as or better than humans and hopefully at a lesser cost. These models are typically trained from a dataset where each example describes an instance of the same task and is represented by a set of characteristics and an expected outcome or label which we usually want to predict. As a practical case, the underlying task can be that of a bank which has to decide whether it should grant a credit to a customer or not. After having collected a dataset describing past credits granted by the bank, a model can be automatically built based on characteristics of the credits (duration, rate, purpose, amount...), of the customers (gender, age, incomes, profession...) and a label stating, *e.g.*, whether or not the customer succeeded in repaying the loan. The model can then be used by the bank to evaluate the risk for new customers applying for a loan to be creditworthy or not. This kind of tasks addressed in this thesis belongs to the supervised classification paradigm. The supervision comes from the fact that we provide to the learning algorithm, in addition to the characteristics, the discrete label of each training example.

All along this document, we will only focus on such supervised classification problems where the number of possible labels is finite. Therefore, we will not address regression (where the number of outcomes is infinite) or unsupervised tasks (where the label is supposed to be unknown at training time, like, *e.g.*, in clustering or dimensionality reduction).

Whatever the learning paradigm, an element shared by the previous settings and required for the success of any machine learning algorithm is related to the quality of the set of characteristics describing the data, also referred as data representation or features. In supervised learning, the more the features describing the examples are correlated with the label, the more effective the model will be. For example, features describing the wages of the customer will probably be more relevant to estimate his/her creditworthiness than the height and weight of the person. Therefore, defining features that capture well these correlations is of high importance in machine learning to build an effective supervised model.

There exist three main families of features. The first one corresponds to the "observable" features which can be easily and directly measured, such as the wages of customers, the blood pressure of a patient, etc. The second category gathers the "handcrafted" features that typically require a certain expertise about the domain, like the SURF [Bay et al., 2006] image descriptors used in computer vision. The last family, we will mainly focus on in this thesis, corresponds to the "latent" features that are usually automatically learned from the training data as done in deep learning [Goodfellow et al., 2016], metric learning [Kulis, 2013, Bellet et al., 2015], PCA, or matrix factorization to cite a few methods. The main advantages of this last category of features (that led to the emergence of the sub-field called representation learning) are the following: *(i)* they allow us to improve the machine learning task by better representing the problem at hand and capturing automatically the correlations with the labels, *(ii)* they allow to overcome the limitations of the handcrafted features that often require a costly human expertise and *(iii)* they are often computationally convenient to process.

The contributions of this thesis fall into the scope of this last category. More precisely, we are interested in the specific setting of learning a discriminative representation when the number of data of interest, often called "positive" examples, is limited. This *de facto* excludes deep learning-based methods which are nowadays those producing indisputably the best performing latent features, but that often require large quantities of training data as well as a tedious tuning process.

A lack of data of interest can be found in different scenarios. The first one arises when the labels of the examples are highly imbalanced, leading to a training dataset where examples of the class of interest are very scarce. This is typically the case in bank fraud detection whose goal consists in detecting fraudulent transactions. In this scenario, the number of frauds is dramatically much smaller than the quantity of genuine transactions. The key issue here is to prevent the learning algorithm from predicting every single example as belonging to the normal class. While such a decision would seem very effective from an accuracy perspective (indeed, more than 99.5% of the transactions are usually genuine), it would definitely fail by missing all the examples of interest (*i.e.*, the frauds).

The scarcity of positive examples also occurs when the data is costly either in terms of money, time or expertise, or even impossible due to the limited history available. This is often the case in the medical domain when one aims at creating a computer aided diagnostic system for a certain disease: doctors often struggle to gather many positive examples due to the cost of capturing features (*e.g.*, MRI scans required to be annotated by experts) or the limited number of past patients, *e.g.*, affected by a rare disease. The peculiarity of this kind of applications prevents us from using standard machine learning methods, like neural networks, which require a large number of examples to learn millions of parameters at the risk of over-fitting the data.

The last scenario finds its roots from the domain adaptation setting [Ben-David et al., 2007, Redko et al., 2019] where the learning algorithm has access to a labeled source dataset as well as target examples with few or even no label. The goal is to benefit from the source to learn an efficient model over the target domain, where the two domains are supposed to be related, yet different. As a practical case, a hospital can use a computer aided diagnostic system for a specific disease developed in another hospital where the characteristics of the patients are measured differently. Due to the differences in the acquisition process (*e.g.*, two different MRI scanners) between the source and target sets, a model learned from the source set will perform poorly on the target set if the discrepancy between the two domains is not taken into account by the model.

To handle these different scenarios, we investigate in this thesis how to learn good representations through the lens of different machine learning frameworks. First, we tackle the problem of imbalanced learning with a class of interest composed of a few examples by learning a metric that induces a new representation space where the learned models do not favor the majority examples. Second, we propose to handle the scenario with few available examples by learning at the same time a relevant data representation and a model that generalizes well through boosting models [Schapire and Singer, 1999] using kernels [Fan et al., 2005] as base learners approximated by random Fourier features [Rahimi and Recht, 2008]. Finally, to address the domain adaptation scenario where the target set contains no label while the source examples are acquired in different conditions, we propose to reduce the discrepancy between the two domains by keeping only the most similar features optimizing the solution of an optimal transport problem [Villani, 2008] between the two domains.

**Context of this thesis.** The work presented in this thesis was carried out in the Data Intelligence team of the Hubert Curien laboratory which is a joint research unit (UMR 5516) between the Jean Monnet University of Saint-Étienne, the University of Lyon, the CNRS and the Institut d'Optique Graduate School. The thesis was financed in part by a ministerial fellowship and by the French project APRIORI ANR-18-CE23-0015.

**Outline of the thesis.** This manuscript is composed of a background chapter followed by three chapters each presenting one of our contributions mentioned above.

- Chapter 1 starts by a general presentation of the supervised classification paradigm which is at the core of this thesis. Then, we present some machine learning algorithms that are used in the following chapters. Finally, we briefly describe some background in metric learning [Kulis, 2013, Bellet et al., 2015], random Fourier features [Rahimi and Recht, 2008] and optimal transport [Villani, 2008] required for the understanding of our contributions.

- Chapter 2 presents our first contribution which consists in the optimization of a metric specifically dedicated to address the challenging problem of learning from highly imbalanced datasets. State-of-the-art metric learning methods usually learn a metric that satisfies a set of constraints which typically aim at moving closer examples of the same class while pushing away examples of different labels. Our proposed method relies on two strategies to deal with the class imbalance: *(i)* a selection of the set of constraints taking into account the class imbalance and *(ii)* a decomposition of the loss into the sum of terms capturing the constraints between examples of different labels. Benefiting from the uniform stability framework [Bousquet and Elisseeff, 2002], we prove a generalization bound that has the main advantage to involve the proportion of rare examples and which encompasses standard metric learning bounds. We show experimentally that our two complementary strategies allow us to reduce the negative impact of an increase in the imbalance.

- Chapter 3 is dedicated to our second contribution. We propose a method that jointly learns a classification model and a representation of the data suited for the classification task at hand and generalizing well in the presence of few labeled examples. Our method leverages two state-of-the-art learning strategies: gradient boosting [Friedman, 2001] to build the classification model, and random features to build the representation. Traditional gradient boosting methods induce an ensemble of regression models by adding one by one to the ensemble the model that best learns the errors made by the previous ensemble. The originality of our contribution comes from the fact that we train at each iteration a kernel defined as a weighted sum of random Fourier features. We show experimentally that the proposed method allows to learn efficiently from non-linearly separable data a compact latent representation.

- Chapter 4 is devoted to the presentation of our third contribution which falls into the scope of domain adaptation where no labeled examples are available from the target domain. The model can benefit from an abundance of labeled data coming from a source domain where the prediction task is the same, but with data acquired in different conditions. To reduce the discrepancy between the two domains, we introduce in this context a feature selection method where the idea is to train a model using the most similar features between the two domains, and to discard the most dissimilar ones. The originality of the contribution is that the similarity of a feature across the two domains is given by the solution of a problem based on the optimal transportation theory. We evaluate our method on different benchmarks and show that selecting the most similar features can improve the performances compared to using all features. We also validate our algorithm on a real world medical imaging task.

# List of Publications

## Publication in Journal

Léo Gautheron, Emilie Morvant, Amaury Habrard and Marc Sebban. Metric Learning from Imbalanced Data with Generalization Guarantees. In *Pattern Recognition Letters*, volume 133, pages 298-304. 2020 [Gautheron et al., 2020c].

## Publications in International Conferences

Léo Gautheron, Pascal Germain, Amaury Habrard, Guillaume Metzler, Emilie Morvant, Marc Sebban and Valentina Zantedeschi. Landmark-based Ensemble Learning with Random Fourier Features and Gradient Boosting. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2020 [Gautheron et al., 2020b].

Léo Gautheron, Amaury Habrard, Emilie Morvant and Marc Sebban. Metric Learning from Imbalanced Data. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019, Portland, United States [Gautheron et al., 2019b].

Léo Gautheron, Ievgen Redko and Carole Lartizien. Feature Selection for Unsupervised Domain Adaptation using Optimal Transport. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2018, Dublin, Ireland [Gautheron et al., 2018b].

## Communications in National Conferences

Léo Gautheron, Pascal Germain, Amaury Habrard, Guillaume Metzler, Emilie Morvant, Marc Sebban and Valentina Zantedeschi. Apprentissage d'ensemble basé sur des points de repère avec des caractéristiques de Fourier aléatoires et un renforcement du gradient. In *Conférence sur l'Apprentissage automatique (CAp)*, 2020 [Gautheron et al., 2020a].

Léo Gautheron, Pascal Germain, Amaury Habrard, Gaël Letarte, Emilie Morvant, Marc Sebban and Valentina Zantedeschi. Revisite des “random Fourier features” basée sur l’apprentissage PAC-Bayésien via des points d’intérêts. In *Conférence sur l’Apprentissage automatique (CAp)*, 2019, Toulouse, France [Gautheron et al., 2019a].

Léo Gautheron, Amaury Habrard, Emilie Morvant and Marc Sebban. Apprentissage de métrique pour la classification supervisée de données déséquilibrées. In *Conférence sur l’Apprentissage automatique (CAp)*, 2018, Rouen, France [Gautheron et al., 2018a].

# List of Notations

| | |
|---|---|
| $\mathbb{N}$ | The set of integers |
| $\mathbb{R}$ | The set of real numbers |
| $\mathbb{R}_+$ | The set of positive real numbers |
| $m, n$ | Number of examples |
| $d$ | Number of features describing each example |
| $c$ | Number of classes |
| $\mathbb{R}^d$ | The set of real-valued vectors composed of $d$ real numbers |
| $\mathbb{R}^{m \times d}$ | The set of real-valued matrices composed of $m$ rows and $d$ columns |
| $\mathcal{X}$ | The input space where the examples of a given task belong and where $\mathcal{X} \subseteq \mathbb{R}^d$ |
| $\mathcal{Y}$ | The output space of a task, considered to be discrete with $\lvert\mathcal{Y}\rvert = c$ |
| $\mathcal{Z}$ | The joint space between inputs and outputs with $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ |
| $\mathcal{D}$ | A distribution over $\mathcal{Z}$ |
| $\boldsymbol{x}$ | An example which is a real-valued vector with $\boldsymbol{x} \in \mathcal{X}$ |
| $y$ | The label of an example with $y \in \mathcal{Y}$ |
| $\mathbf{z}$ | A labeled example such that $\mathbf{z} = (\boldsymbol{x}, y)$ and $\mathbf{z} \in \mathcal{Z}$ |
| *i.i.d.* | Independently and identically distributed |
| $\mathbf{z} \sim \mathcal{D}$ | The *i.i.d.* draw of a labeled example $\mathbf{z}$ according to the distribution $\mathcal{D}$ |
| $S$ | A set of $m$ labeled examples with $S = \{\mathbf{z}_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ |
| $S \sim \mathcal{D}^m$ | The *i.i.d.* draw of $m$ labeled examples according to $\mathcal{D}$ |
| $\boldsymbol{X}^S, \boldsymbol{Y}^S$ | The examples and labels of $S$ such that $\boldsymbol{X}^S = \{\boldsymbol{x}_i\}_{i=1}^{m}$ and $\boldsymbol{Y}^S = \{y_i\}_{i=1}^{m}$ |
| $\mathbf{z}_i$ | The $i$th labeled example in a set of examples with $i \in \{1, \ldots, m\}$ |
| $x_i$ | The $i$th real value (or feature) of an example $\boldsymbol{x}$ with $i \in \{1, \ldots, d\}$ and $x_i \in \mathbb{R}$ |
| $\boldsymbol{X}^{S\top}$ | The transpose of $\boldsymbol{X}^S \in \mathbb{R}^{m \times d}$ such that $\boldsymbol{X}^{S\top} \in \mathbb{R}^{d \times m}$ |
| $[u, v]$ | A continuous interval of real numbers that includes both $u \in \mathbb{R}$ and $v \in \mathbb{R}$ |
| $\{u, \ldots, v\}$ | A discrete interval of integers going from $u \in \mathbb{N}$ to $v \in \mathbb{N}$ by increment of 1 |
| $\{u, v, w\}$ | A set of values containing only the elements listed |

# Chapter 1

# Background

**Abstract**

We give in this chapter a general presentation of the supervised classification paradigm which is at the core of this thesis. Then, we present some machine learning algorithms that are used in the following chapters. Finally, we briefly describe some background in metric learning [Kulis, 2013, Bellet et al., 2015], random Fourier features [Rahimi and Recht, 2008] and optimal transport [Villani, 2008] required for the understanding of our contributions.

## 1.1 Supervised learning

### 1.1.1 Learning from labeled data

Throughout this thesis, we consider supervised tasks where the training examples are $d$-dimensional real-valued vectors each belonging to an input space $\mathcal{X} \subseteq \mathbb{R}^d$ and are assigned a class label coming from a discrete output space $\mathcal{Y}$ such that $|\mathcal{Y}| = c$. We mainly consider in this thesis the binary classification setting where $c = 2$ and $\mathcal{Y} = \{-1, +1\}$. Together, the spaces $\mathcal{X}$ and $\mathcal{Y}$ form a joint space noted $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.

When training a model, the supervised learning algorithms are provided with a training set composed of $m$ labeled examples $S = \{\mathbf{z}_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^m$ where $\forall i \in \{1, \dots, m\}$, $\boldsymbol{x}_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$ and $\mathbf{z}_i \in \mathcal{Z}$. We further define the set of examples as the matrix $\boldsymbol{X}^S = \{\boldsymbol{x}_i\}_{i=1}^m$ such that $\boldsymbol{X}^S \in \mathbb{R}^{m \times d}$ and the set of labels as $\boldsymbol{Y}^S = \{y_i\}_{i=1}^m$. We suppose that these $n$ examples are *i.i.d.* according to an unknown distribution $\mathcal{D}$ over $\mathcal{Z}$ and we note this *i.i.d.* draw of $m$ elements by $S \sim \mathcal{D}^m$.

Given a predictive task and an associated training set $S$ drawn according to the distribution $\mathcal{D}$, the aim of supervised learning [Bishop, 2006] is to find using $S$ a model that produces correct predictions for the examples drawn according to $\mathcal{D}$.

We consider models that are functions of the form $h : \mathcal{X} \to \mathcal{L}$ with $\mathcal{L}$ is the output set which is mainly induced by the learning algorithm used and often $\mathcal{L} \subseteq \mathbb{R}$. Each function $h$ belongs to a family of functions $\mathcal{H}$ that depends on the learning algorithm used. Inside a family $\mathcal{H}$, we denote by $\mathbf{a}$ the parameters that characterize the model $h$ and differentiate it from the other models in $\mathcal{H}$, and we refer to the model $h$ with its parameters as $h_{\mathbf{a}}$. Thus when searching for a model producing correct predictions, we are actually searching inside a family of functions $\mathcal{H}$ the parameters $\mathbf{a}$ inducing the model $h_{\mathbf{a}}$ that handles the best the predictive tasks given. By abuse of notation, we often refer as the trained model and its parameters as $h$ instead of $h_{\mathbf{a}}$.

The output set $\mathcal{L}$ of the model $h$ might be equal to the output space $\mathcal{Y}$ or not. For example in binary classification tasks, we can have $\mathcal{L} = \mathbb{R}$ where the sign of the prediction indicates the class $+1$ or $-1$, and the larger the absolute value of the prediction, the larger the confidence in the prediction. Similarly, we can encounter $\mathcal{L} = [0, 1]$ when the prediction corresponds to the probability of belonging to a class. For classification tasks with $c > 2$, we often have $\mathcal{L}$ which is the set of vectors of size $c$ where each element is a membership probability.

After a model has been trained, it can be used to make predictions on new examples supposed to be also *i.i.d.* from the same distribution $\mathcal{D}$. The aim is that the predictions of the learned model $h$ will not only be accurate on the examples in $S$, but also for any $(\boldsymbol{x}, y) \sim \mathcal{D}$.

Usually, a smaller set of $n$ labeled examples $T = \{\mathbf{z}_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, called test set, is kept aside during the training but used by the trained model to make predictions and compare them with the true labels $\boldsymbol{Y}^T$ in order to measure how well the model performs on new examples not seen during the training. It is known that such a way to proceed allows us to get an unbiased estimate of the generalization ability of the model. Figure 1.1 gives a graphical illustration of the different notations introduced. It shows a model that perfectly predicts the labels of a set of examples. In practice, however, a model rarely produces perfect predictions. In order to compare candidate models, we need performance measures.

### 1.1.2 Performance measures

Given a trained model $h$ and a set of $n$ examples $T$ with labels $\boldsymbol{Y}^T \in \mathcal{Y}^n$, we define $\widehat{\boldsymbol{Y}}^T \in \mathcal{Y}^n$ the set of predicted labels over $T$ using $h$. The goal of a performance measure is to compare the predicted labels in $\widehat{\boldsymbol{Y}}^T$ with the actual labels in $\boldsymbol{Y}^T$ and thus evaluate how good the predicted labels are. In a classification setting, a usual performance measure is the accuracy, defined as

$$\begin{aligned}\text{accuracy} =& \frac{\text{Number of correctly classified examples}}{\text{Total number of examples}}\\ =& \frac{\sum_{i=1}^{n}\left[\widehat{Y}_i^T = Y_i^T\right]}{n},\end{aligned}$$

where [.] is an indicator function equal to either 0 or 1. The accuracy takes a value in $[0, 1]$ with a higher value indicating more accurate predictions.

In this thesis, we often deal with binary classification tasks with $\mathcal{Y} = \{-1, +1\}$ where an example is either of the positive class (+1) or the negative class (-1). In such a context, we can define other performance measures based on the four quantities described in the confusion matrix of Table 1.1. The accuracy can be rewritten with these notations as

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}.$$

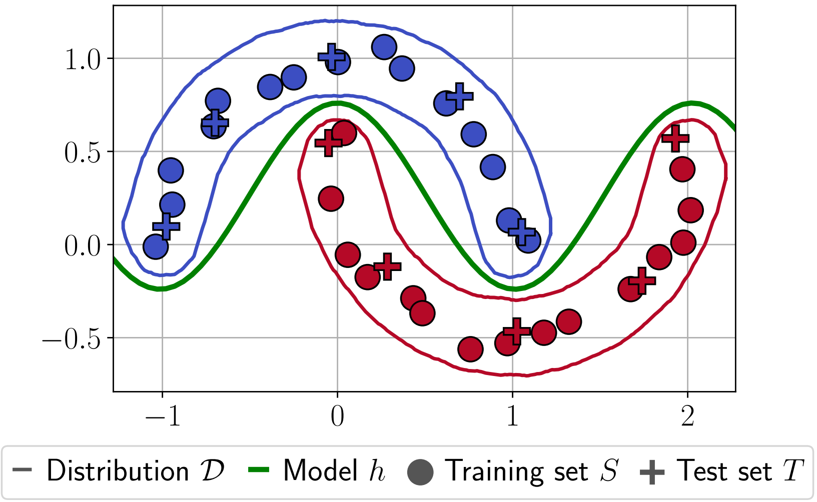


*Figure 1.1: Illustration of a toy dataset divided into $S$ with $m = 30$ (circles) and $T$ with $n = 10$ (crosses). Each example $\boldsymbol{x}$ is described by $d = 2$ features (the real-values along the two axes noted respectively $x_1$ and $x_2$) and by a label among $\mathcal{Y} = \{blue, red\}$. $S$ and $T$ are supposed to be drawn according to a distribution $\mathcal{D}$ represented in this toy example by the blue and red concave hulls. Here, the goal is to learn a model using $S$ that can predict the color of a point given its $x_1$ and $x_2$ coordinates. Ideally, this model should predict correctly the label of any point drawn from $\mathcal{D}$ (here drawn inside the two concave hulls). Here, we use the family of models $\mathcal{H}$ of the form $h_{\mathbf{a}=(\theta_1,\theta_2,\theta_3)} : x \to (\theta_1 + \theta_2 \cos(\theta_3 x))$ and select by hand the model with parameters* $\mathbf{a} = (0.26, 0.5, 3.1)$*. As the predictions produced by $h_{\mathbf{a}}$ are in $\mathcal{L} = \mathbb{R}$, they can be converted to a value in $\mathcal{Y}$ by the rule “IF $x_2 < h_{\mathbf{a}}(x_1)$ THEN red ELSE blue”. This model allows to predict with no mistakes the labels of all examples in $T$ and any possible example drawn according to $\mathcal{D}$.*

In imbalance learning, where the number of positive examples ($y = +1$) is much smaller than the number of negatives ($y = -1$), the accuracy is not well suited to evaluate a model. Indeed, a function predicting all examples as negative would have an accuracy close to 1 while definitely missing the examples of the class of interest. Because of this, other performance measures are available focusing more on the minority class. The first one called the precision is defined as

$$\text{precision} = \frac{TP}{TP + FP},$$

and describes how well a model is *precise on the positive examples* by giving among the predicted positive examples the proportion that are actually positive. A second one called the recall is defined as

$$\text{recall} = \frac{TP}{TP + FN}.$$

It describes how well a model behaves to *retrieve the positive examples* by giving among the actual positive examples the proportion that is recovered.

Maximizing either the precision or recall alone is not enough as it tends to decrease the other one. Instead, a trade-off between the two is preferred, such as the F$\beta$-measure [Van Rijsbergen,

*Table 1.1: Confusion matrix in binary classification tasks. By comparing the actual labels $\boldsymbol{Y}^T = \{y_i\}_{i=1}^n$ with their corresponding predicted labels $\widehat{\boldsymbol{Y}}^T = \{\widehat{y}_i\}_{i=1}^n$ we can define four quantities called TP, FN, FP and TN that count the number of example in each of the four possible cases such that $TP + FN + FP + TN = n$.*

| | Predicted positives $\widehat{y} = +1$ | Predicted Negatives $\widehat{y} = -1$ |
|---|---|---|
| Positive examples $y = +1$ | True Positives (TP) | False Negatives (FN) |
| Negative examples $y = -1$ | False Positives (FP) | True Negatives (TN) |

1974] defined as

$$\text{F}\beta\text{-measure} = \frac{(1+\beta^2)\text{precision} \times \text{recall}}{(\beta^2\text{precision}) + \text{recall}},$$

where $\beta > 1$ gives more weight to the recall and $0 < \beta < 1$ gives more weight to the precision. Note that $\beta = 1$ gives an equal weight to both quantitie and leads to the well known F1-measure:

$$\text{F1-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.$$

Another performance measure taking into account the imbalance in the data is the Area under the ROC Curve (AUC). It is based on the recall and on the false positive rate defined as

$$\text{false positive rate} = \frac{FP}{FP + TN}.$$

Note that unlike the previous performance measures that are computed using only $\boldsymbol{Y}^T$ and $\widehat{\boldsymbol{Y}}^T$, the AUC requires to be computed with $\boldsymbol{Y}^T$ and a set of different predictions for different values of a threshold $\tau \in \mathcal{L}$ defined as $\widehat{\boldsymbol{Y}}_\tau^T = \big\{ +1 \text{ IF } h(\boldsymbol{x}_i) \geq \tau \text{ OTHERWISE} - 1\ \forall i \in \{1, \dots, n\}\big\}$. The AUC is then computed as the area under the ROC curve defined for different values of $\tau$ and associated predictions $\widehat{\boldsymbol{Y}}_\tau^T$ as $f(\text{false positive rate}) = \text{recall}$.

### 1.1.3 Loss functions

To guide the search for the parameters $\mathbf{a}$ inducing a model $h_{\mathbf{a}} \in \mathcal{H}$ having a good performance measure, machine learning algorithms resort to loss functions $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}_+$ that measure how much wrong a model is at predicting the value of a labeled example $\mathbf{z} \in \mathcal{Z}$. In the case of binary classification with $\mathcal{L} = \mathbb{R}$, the loss function that is usually minimized is called the zero-one loss defined as

$$\ell(h, \mathbf{z}) = \big[\text{sign}\big(h(\boldsymbol{x})\big) \neq y\big].$$

Minimizing the zero-one loss is equivalent to maximizing the accuracy. However, like the other performance measures introduced in the previous section, this loss is NP-hard to optimize. Instead, surrogate loss functions of the zero-one loss that are both convex and differentiable can be used. For example, the exponential loss defined as follows

$$\ell(h, \mathbf{z}) = \exp\big(-yh(\boldsymbol{x})\big),$$

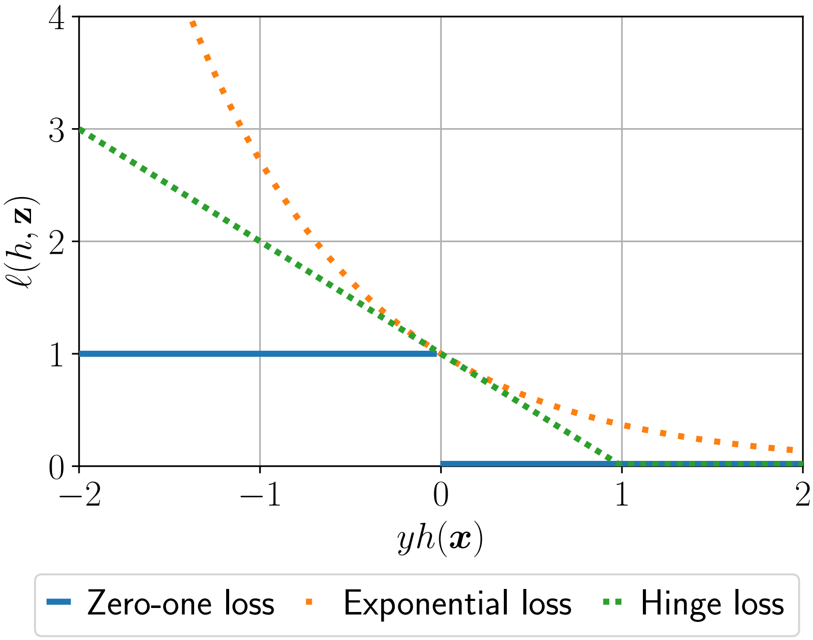


*Figure 1.2: Values of the zero-one loss and two of its surrogates for an example $\mathbf{z} = (\boldsymbol{x}, y)$ and a model $h$ in function of the prediction $h(\boldsymbol{x}) \in \mathbb{R}$ multiplied by $y \in \{-1, +1\}$.*

is often used in boosting [Freund and Schapire, 1996] and it takes into account the disagreement between the prediction $h(\boldsymbol{x})$ and the actual label $y$. We can also cite the hinge loss defined as

$$\begin{aligned}\ell(h, \mathbf{z}) =& [1 - yh(\boldsymbol{x})]_+\\ =& \max\big(0, 1 - yh(\boldsymbol{x})\big),\end{aligned}$$

often used with Support Vector Machines (SVM) [Vapnik, 1995] and which penalizes predictions of $h(\boldsymbol{x})$ having a different sign than $y$ or having a confidence smaller than 1. We show the behavior of these three losses in Figure 1.2.

Finally, we mention the least square loss defined as follows:

$$\ell(h, \mathbf{z}) = \big(y - h(\boldsymbol{x})\big)^2,$$

often used for regression tasks and in particular in gradient boosting [Friedman, 2001] and penalizes predictions $h(\boldsymbol{x})$ far from the label $y \in \mathbb{R}$.

Computing a loss over all the distribution $\mathcal{D}$ leads to the true risk $R(h)$ of a model and is defined as the following expected value:

$$R(h) = \mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{D}} \ell(h, \mathbf{z}).$$

Given a loss $\ell$, even if the goal is to select the model $h^* \in \mathcal{H}$ leading to the smallest true risk, in general we cannot find $h^*$ based only on the quantity $R(h)$ because we cannot compute it as $\mathcal{D}$ is unknown. What is done in practice is rather to minimize the empirical risk $\widehat{R}(h)$ defined for a training set $S$ as the empirical mean of the loss:

$$\widehat{R}(h) = \frac{1}{m}\sum_{i=1}^{m} \ell(h, \mathbf{z}_i).$$

Minimizing the empirical risk might be insufficient to obtain a good model, especially when the number of training example is small. For example, a trivial model $h$ that would memorize all the training examples would have $\widehat{R}(h) = 0$ while being potentially wrong on many unseen examples. This phenomenon is called over-fitting and it can be avoided in practice by learning the parameters $\mathbf{a}$ of a model $h$ that minimize a trade-off between the empirical risk and a regularization term on $\mathbf{a}$:

$$\arg\min_{\mathbf{a}} \widehat{R}(h_{\mathbf{a}}) + \lambda Reg(\mathbf{a}), \tag{1.1}$$

where $Reg(\mathbf{a}) > 0$ becomes larger when the complexity of the model $h_{\mathbf{a}}$ increases, *i.e.*, when the model tends to over-fit. The aim of the regularization is to prevent to obtain too complex models by imposing constraints on the parameters learned. In this case, the impact of the regularization on the solution is controlled by a hyper-parameter $\lambda > 0$. A value of $\lambda$ too close to 0 may lead to a model that tends to over-fit, while a too large value of $\lambda$ may lead to a model that under-fits, *i.e.*, that has a large empirical risk. In general, it is not appropriate to learn the weight of the regularization at the same time as the model parameters $\mathbf{a}$ to minimize Equation (1.1) because doing so would always yield $\lambda = 0$ as best minimizer. Instead, the weight of the regularization can be tuned in a different learning step described in the following.

### 1.1.4 Parameter tuning

If a model $h$ is trained with regularization to reduce its complexity, a hyper-parameter is used to control the trade-off between the capacity of $h$ to fit the training data and the complexity of $h$ which is related to its generalization capacity. A possibility to tune $\lambda$ is *(i)* to train several models from $S$ but for different values of the hyper-parameter, and then *(ii)* to select the model and the corresponding $\lambda$ having the more accurate predictions on $T$. However, $T$ is usually kept aside to evaluate the model when all parameters are learned, simulating how it performs on examples never seen before. Instead, this hyper-parameter can be tuned through a process called cross-validation using only $S$.

The idea of cross-validation depicted in Figure 1.3 is first to partition $S$ into a number of disjoint subsets called folds. Then for each fold, a model is trained on the union of the remaining subsets and used to compute a performance measure describing how correct its predictions on this fold are. Finally, the hyper-parameter leading to the best performance measure averaged over the folds can be selected to retrain the model on the entire set $S$. This model can latter be used to compute the same performance measure on $T$, giving an estimate of the true risk.

Note that with more than one hyper-parameter, each having its own set of possible values, it may be necessary to perform a grid-search during the cross-validation because changing the value of a hyper-parameter may change the best value of another. The grid-search consists in repeatedly learning and evaluating the model with cross-validation for every possible combination of hyper-parameters/values. This becomes rapidly expensive in computation time when increasing the number of hyper-parameters and their numbers of possible values, especially when the model requires itself a large amount of training time. Even so, the cross-validation

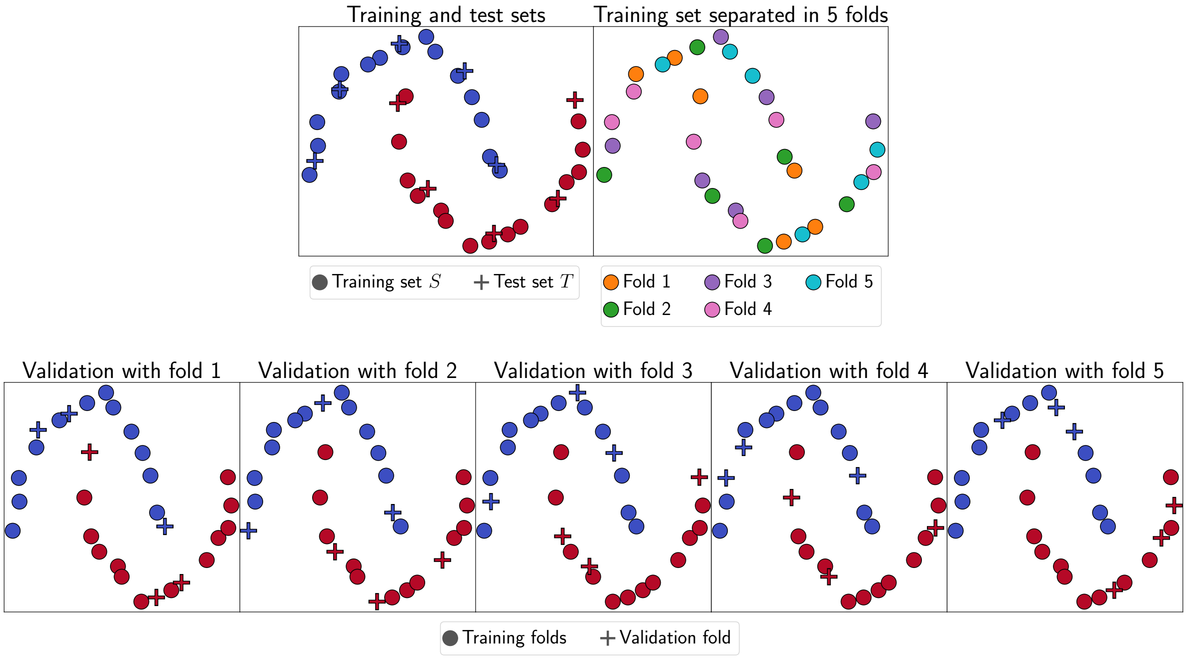


*Figure 1.3: Illustration of the cross-validation process. The first plot in the first row shows a dataset divided into a training and a test set. The second plot in the first row shows the first cross-validation step consisting in partitioning $S$ in subsets called folds, here in 5. Then in the second row, for each fold, a model is trained on the assembled remaining folds and used to compute a performance measure describing how correct its predictions on this fold are. Finally the performance averaged over the folds can be used as a parameter tuning criterion.*

step is useful in practice for methods having few hyper-parameters to tune.

### 1.1.5 Generalization guarantees

We described in the previous sections how supervised models can be trained and evaluated using different set of examples drawn according to the same unknown distribution $\mathcal{D}$. However, the quality of the evaluation is limited due to the fact that both $S$ and $T$ are finite. Ideally, we would like to give guarantees that the performance on the training set will be close to the performances on all the possible examples drawn according to $\mathcal{D}$. During the last decades, several theories [Bartlett and Mendelson, 2002, Bousquet and Elisseeff, 2002, McAllester, 1999, Valiant, 1984, Vapnik and Chervonenkis, 1971] have been developed to analyze under which conditions these quantities are close based on the widely used Probably Approximately Correct framework [Valiant, 1984]. This can be done by upper-bounding by a value $\epsilon \geq 0$ the deviation between the empirical risk and the true risk with a probability at least $1-\delta$ with $\delta \in [0,1]$ over the random draw of $m$ examples according to $\mathcal{D}$:

$$\underset{S\sim\mathcal{D}^m}{\mathbb{P}}\left(\left|\widehat{R}(h) - R(h)\right| \leq \epsilon\right) \geq 1-\delta,$$

or in a less restrictive way

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\Big(R(h) \le \widehat{R}(h) + \epsilon\Big) \ge 1 - \delta.$$

Whatever the theoretical framework, the bounds typically behave as follows: *(i)* the more confident we are (*i.e.*, $\delta$ tends to 0), the looser the bound (*i.e.*, $\epsilon$ tends to $+\infty$) and *(ii)* the bound becomes tighter ($\epsilon$ tends to 0) when $m$ the number of training examples increases.

We now specifically present two theories: the uniform stability framework [Bousquet and Elisseeff, 2002] which we use to derive guarantees for our contribution in Chapter 2, and the PAC-Bayesian theory [McAllester, 1999] that motivates our contribution in Chapter 3.

**Uniform Stability** The uniform stability framework [Bousquet and Elisseeff, 2002] allows one to derive generalization bounds for learning algorithms taking the form of a minimization trade-off between a convex loss function and a regularization. The bounds derived through this framework have the advantage to take into consideration properties of the learning algorithm such has its hyper-parameters, its regularization and its loss function. This framework can be used to derive bounds for algorithms that have the following property called uniform stability.

**Definition 1** ([Bousquet and Elisseeff, 2002] definition 6)**.** *Given a distribution $\mathcal{D}$, a learning algorithm has uniform stability $\beta \ge 0$ with respect to a loss $\ell$ if $\forall S \sim \mathcal{D}^m$ and $\forall i \in \{1,\dots,m\}$ the following holds*

$$\sup_{\mathbf{z}\in S} \left|\ell(h,\mathbf{z}) - \ell(h^i,\mathbf{z})\right| \le \beta$$

*where $h$ is the model learned with the algorithm from $S$, and $h^i$ is the model learned with the algorithm from $S^i$, the set obtained by replacing the $i^{th}$ example in $S$ by another also i.i.d. from $\mathcal{D}$.*

This property tells us that the deviation of the loss on the training examples between $h$ and $h^i$ is upper bounded by a value $\beta$. The intuition is that learning a second model $h^i$ after a small modification of the training set gives almost the same model as $h$, where the difference between the two models is quantified by $\beta$. This value takes into account the regularization weight of the algorithm and the number of examples in the training set, where a larger regularization hyper-parameter and a larger number of examples allow to obtain a smaller $\beta$. And the smaller $\beta$, the more precise the resulting bounds thanks to the following theorem:

**Theorem 1** ([Bousquet and Elisseeff, 2002] Th. 12)**.** *Consider a learning algorithm having stability $\beta$ with respect to a loss $\ell$ such that $\forall S \sim \mathcal{D}^m$ and $\forall \mathbf{z} \in S$ then $0 \le \ell(h,\mathbf{z}) \le M$ where $h$ is the model learned from $S$. Then for any $m \ge 1$ with a probability at least $1-\delta$ over the random choice of $S \sim \mathcal{D}^m$, we have the following bound on the true risk $R(h)$:*

$$R(h) \le \widehat{R}(h) + 2\beta + \Big(4m\beta + M\Big)\sqrt{\frac{\ln(1/\delta)}{2m}}.$$

Unlike Vapnik-Chervonenkis dimension-based bounds [Vapnik and Chervonenkis, 1971], the uniform stability framework allows us to derive guarantees even for family of hypotheses of infinite VC-dimension, by taking into account the properties of the algorithm. We will make use of this setting in Chapter 2 to derive guarantees on our metric learning algorithm devoted to address imbalanced learning problems.

**PAC-Bayesian theory** This theory [McAllester, 1999, Shawe-Taylor and Williamson, 1997] allows one to derive generalization bounds for models defined as weighted majority votes over a family of models $\mathcal{H}$. To do so, two weighting distributions over $\mathcal{H}$ are considered. The first one called prior distribution, noted $p$, gives a prior knowledge (before observing a training set) on which of the models in $\mathcal{H}$ are better than the others to handle the task (it can be the uniform distribution is the absence of prior knowledge). The second one called posterior distribution, noted $q$, is learned from a training set of examples, and is then used to define the following majority vote model for binary classification:

$$B_q(\boldsymbol{x}) = \operatorname{sign}\Big(\mathop{\mathbb{E}}_{h\sim q} h(\boldsymbol{x})\Big).$$

Usually, $q$ is learned to minimize an upper bound on the true risk $R(B_q)$ with respect to the zero-one loss. Directly minimizing $R(B_q)$ is difficult. Instead, one can obtain an indirect bound on $R(B_q)$ by upper-bounding the so-called risk of the Gibbs classifier, noted

$$R(G_q) = \mathop{\mathbb{E}}_{h\sim q} R(h).$$

Indeed, one can relate the two with the following inequality (see Langford and Shawe-Taylor [2003] Lemma 4.1):

$$R(B_q) \leq 2R(G_q).$$

Then, one can use the following generalization bound on $R(G_q)$ derived from McAllester [1999]:

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left(\forall q \text{ on } \mathcal{H},\ R(G_q) \leq \widehat{R}(G_q) + \sqrt{\frac{\mathrm{KL}(q\|p) + \ln\frac{2\sqrt{m}}{\delta}}{2m}}\right) \geq 1-\delta,$$

where $\mathrm{KL}(q\|p) = \mathop{\mathbb{E}}_{h\sim q} \ln\frac{q(h)}{p(h)}$ is the Kullback-Leibler divergence between the two distributions $q$ and $p$. There exists numerous other types of PAC-Bayesian bounds, but this one is interesting as it allows one (see Germain et al. [2009]) to compute for any family of classifiers $\mathcal{H}$, any prior $p$ any regularization parameter $\beta > 0$ to tune, the minimizer $q^*$ of the bound with the following closed-form equation

$$q^*(h) = \frac{1}{Z}p(h)\exp\left(-\beta\widehat{R}(h)\right),$$

where $Z$ is a normalization constant. This bound and its minimizer are useful for us because we base our contribution in Chapter 3 on a specific PAC-Bayesian analysis inspired from this setting.

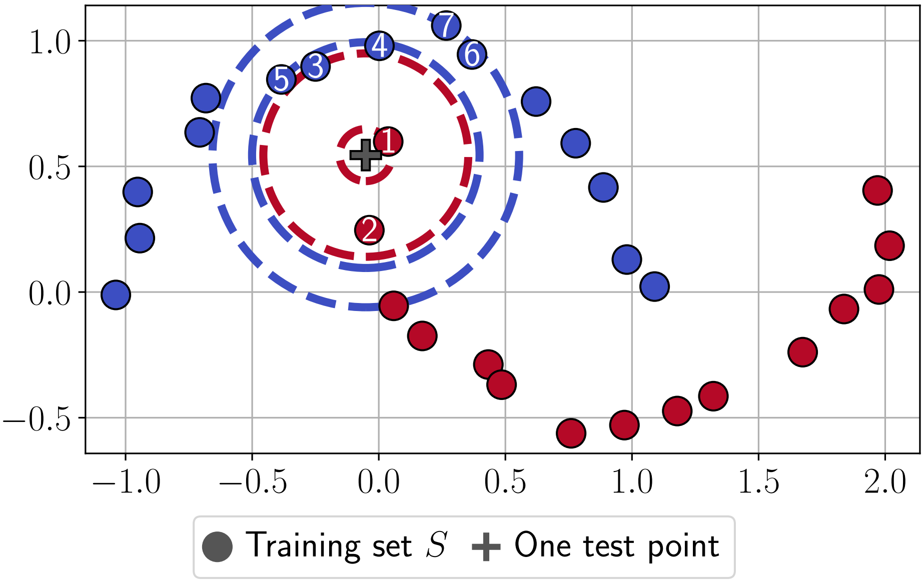


*Figure 1.4: Decision rule for the k-Nearest Neighbor algorithm. We display for one test point (the black cross in the figure) how its predicted color (red or blue) is selected among colored dashed circles around the point enclosing a varying number of neighbors $k \in \{1, 3, 5, 7\}$. The closest training points from the test point are numbered from 1 (closest) to 7 with respect to the Euclidean distance. We see that with $k \in \{1, 3\}$ the test point is classified as red, and with $k \in \{5, 7\}$ the test point is classified as blue.*

## 1.2 Classification algorithms

In this section, we present five classification algorithms that will be used throughout the rest of this thesis.

### 1.2.1 k-Nearest Neighbor (kNN)

This algorithm [Cover and Hart, 1967] does not involve the usual first step devoted to learning a set of learnable parameters $\mathbf{a}$ as it has no parameter to learn. That is why kNN belongs to the so-called lazy algorithms. Instead, this algorithm directly produces predictions based on a training set $S$, a distance function and an integer $k$. Predicting the class of an example $\boldsymbol{x}$ requires to compute the distance between $\boldsymbol{x}$ and all the $m$ examples in $S$ and to retain the $k$ closest examples in $S$. The algorithm then returns as prediction for $\boldsymbol{x}$ the most represented class among these $k$ nearest neighbors. We show in Figure 1.4 how a class prediction is made in function of an increasing value of $k$. Figure 1.5 depicts decision boundaries produced for different values of $k$. Note that we can see $k$ as a hyper-parameter to tune which regularizes in some way the classifier. Indeed, a small value tends to over-fit the data (the decision is made very locally), while a large value tends to under-fit (the classifier tends to predict the majority class).

To avoid having ties when selecting the most represented class, a possibility in binary

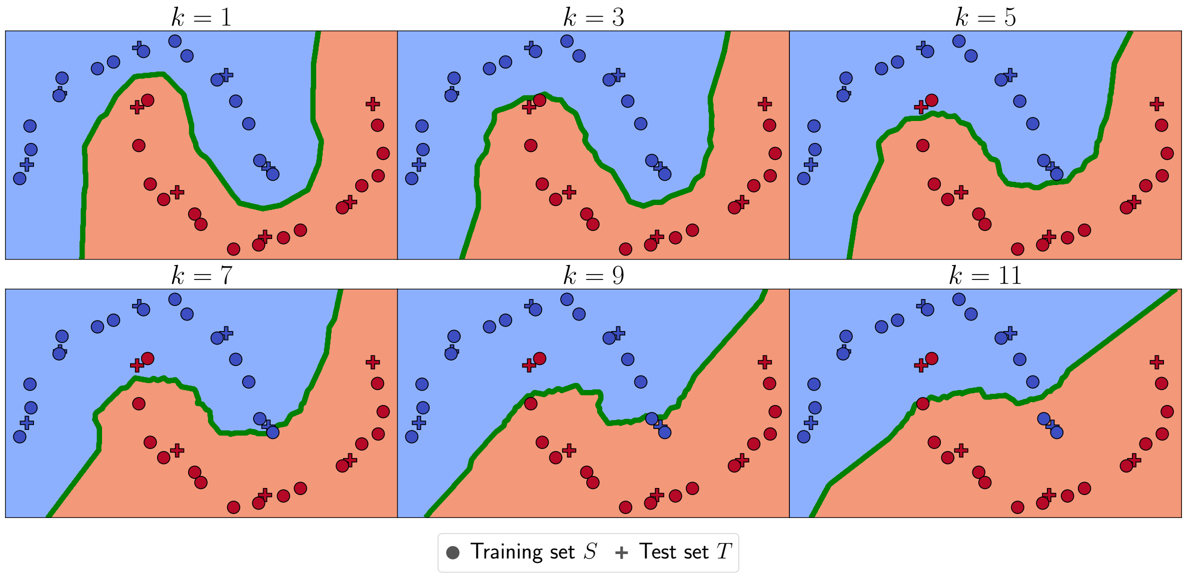


*Figure 1.5: Decision boundary for the k-Nearest Neighbor algorithm for different values of k. A point falling in a red (resp. blue) area is classified as red (resp. blue). Here, by increasing k the model tends to under-fit the data as some training example become incorrectly classified.*

classification is to limit ourselves to odd values of $k$. When dealing with multi-class problems, a solution [Dudani, 1976] is to give a weight to each neighbor inversely proportional to its distance, and to choose the class having the largest sum of weights.

A key component of the kNN algorithm is the distance function used to compute the distance between the examples. This distance can be learned with the help of metric learning algorithms [Kulis, 2013, Bellet et al., 2015] to build new distance functions more suited than the Euclidean distance. Our contribution in Chapter 2 is based on this idea, and we will show how to learn an effective metric in the presence of few positive examples.

### 1.2.2 Support Vector Machine (SVM)

SVM [Boser et al., 1992, Cortes and Vapnik, 1995] is a learning algorithm that builds hyper-planes that can be used to classify examples depending on which of the two sides they are with the following rule:

$$h(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b),$$

where the sign of $(\boldsymbol{w} \cdot \boldsymbol{x} + b)$ indicates which side $\boldsymbol{x}$ is from the hyper-plane, and the absolute value indicates how far it is. The parameters $\boldsymbol{w}$ and $b$ define the hyper-plane and are learned to optimize a regularization/risk trade-off defined as follows:

$$\begin{aligned} \min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{i=1}^{m} \xi_i \qquad (1.2) \\ \text{subject to} \quad & y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad \forall i \in \{1, \dots, m\}, \end{aligned}$$

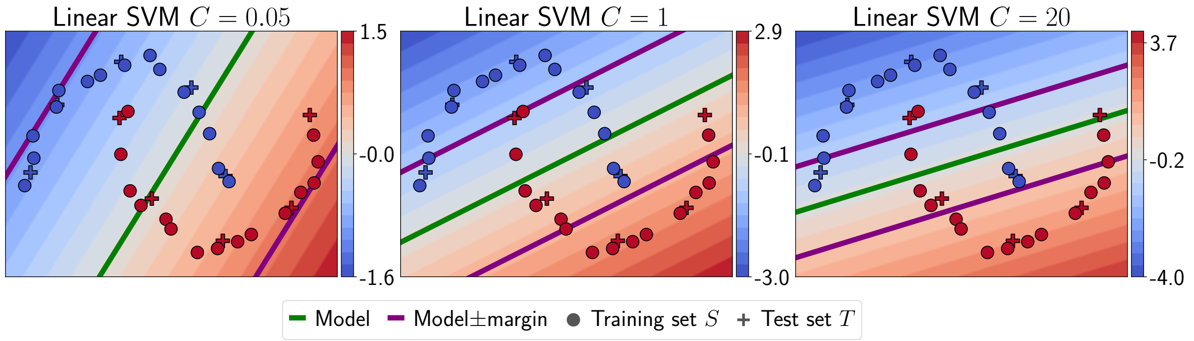


*Figure 1.6: SVM model for different values of the parameter $C \in \{0.05, 1, 20\}$. A value of $C$ close to 0 privileges models with a large margin but with the downside of having more examples violating the constraint of being on the correct side of the hyper-plane at a distance larger than the margin. A large value of $C$ concentrates on minimizing the number of violated constraints and has for effect to decrease the safety margin. The blue (resp. red) examples are assigned the label $-1$ (resp. $+1$), and the SVM model returns predictions in $\mathbb{R}$ (in $[-4, 3.7]$ with $C = 20$) where a negative (resp. positive) prediction corresponds to the blue (resp. red) class.*

where $\xi_i$ are slack variables and where *(i)* the examples are constrained to be on the correct side of the hyper-plane (risk minimization) with the soft constraint $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i$ with $\sum_{i=1}^m \xi_i$ as small as possible, and *(ii)* the examples are constrained to have a distance to the hyper-plane greater than a value called margin that is maximized (regularization) by minimizing the norm of $\boldsymbol{w}$. As shown in Figure 1.6, a large value of $C > 0$ tends to over-fit by making the soft constraint hard, while a $C$ close to 0 allows to produce more regularized models with a large margin.

Problem 1.2 is often called primal problem. In practice, we rather resort to the optimization of its dual counterpart for computational reasons and because the dual allows to use kernel functions. The dual problem is defined as follows:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m y_i y_j \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^m \alpha_i \\ \text{subject to} \quad & \sum_{i=1}^m y_i \alpha_i = 0, \\ \text{and} \quad & 0 \leq \alpha_i \leq C, \quad \forall i \in \{1, \dots, m\}, \end{aligned}$$

where $\boldsymbol{\alpha}$ can be learned efficiently by updating iteratively only two carefully selected values in the vector (to enforce the constraints) until convergence [Fan et al., 2005].

The function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called a kernel, and it measures a similarity between its two arguments. Using this formulation, the decision function is defined as

$$h(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^m y_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b\right),$$

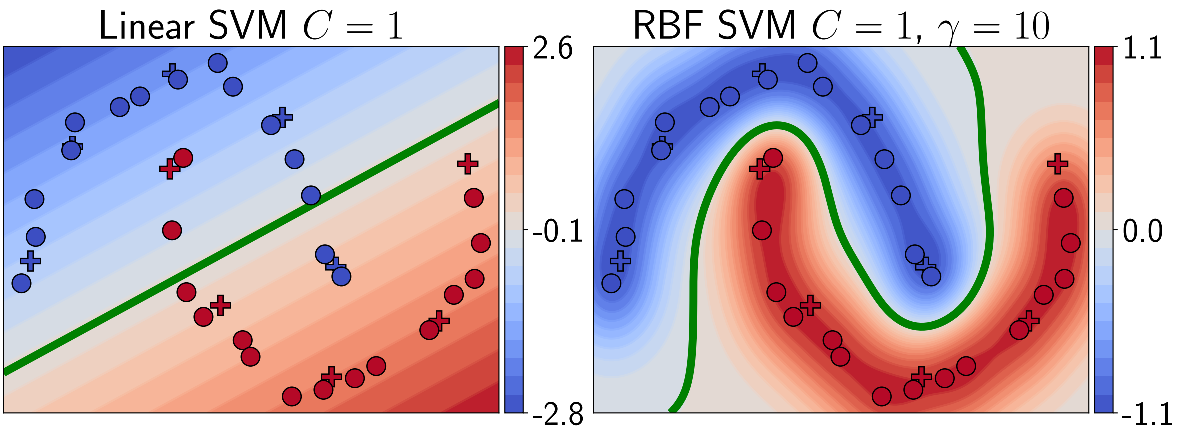


*Figure 1.7: SVM model for a linear and an RBF kernel. The linear kernel allows to build a linear separator. The RBF kernel enables to learn a linear separator in a higher dimensional representation space which induces a non linear separator in the initial space.*

where by plugging the linear kernel defined as the scalar product between two points

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x} \cdot \boldsymbol{x}',$$

we obtain the same formulation as the one from the primal problem with

$$\boldsymbol{w} = \sum_{i=1}^{m} y_i \alpha_i \boldsymbol{x}_i.$$

Using this formulation, $b$ is not learned but is deduced from $\boldsymbol{\alpha}$ (see [Chang and Lin, 2011]) as

$$b = -\frac{\sum_{i:0<\alpha_i<C}^{m} y_i \nabla_i f(\alpha)}{|\{i|0 < \alpha_i < C\}|},$$

where $\nabla_i f(\boldsymbol{\alpha})$ is the $i$th component of the partial derivative of $f$ with respect to $\boldsymbol{\alpha}$.

The use of kernel functions different from the linear kernel is interesting because it allows to compare points in a different representation space without requiring to compute explicitely the (costly) projection of the points in that space [Mercer, 1909]. This cheap computation of the scalar product through a kernel function is known as “kernel trick”. For example, the RBF kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|^2),$$

induces a representation space with an infinite number of dimensions, with $\gamma > 0$ a parameter to tune. For this kernel, the “kernel trick” is useful as otherwise it would be impossible to first project the points in the space before computing their scalar products. Instead, the use of the formula of the RBF kernel is cheap to compute and is equivalent to the scalar product between the points in the infinite dimensional space. Furthermore, when using the kernel function to compare the points in the dual formulation of the SVM, this allows to build a linear separator in the representation space induced by the kernel. The advantage is that if the examples are not linearly separable in their original representation, they may be separable in the space induced by the kernel. This behavior is shown in Figure 1.7 where a linear separator in the original space fails to separate the points using the linear kernel. On the other hand, with the RBF kernel, the linear separator in the space induced by the kernel becomes a non-linear separator in the original space that successfully separates the points.

### 1.2.3 Decision tree

In machine learning, decision trees are models made of a set of nodes connected between each other through parent/child connections. These trees usually starts at a unique node called root that has no parent, and ends at different nodes called leaves that have no child. In classification, each leaf corresponds to a single class and all examples falling in a leaf are predicted as belonging to its corresponding class. In regression, a leaf gives as output the mean target value of its containing training examples. Current implementations of decision trees are often based on the work of Breiman et al. [1984] where the trees are binary, meaning that each node except the leaves has two child, and that an example is assigned to the left or right child depending if a test on a single feature of this example is verified or not.

The complexity of the trees can be controlled by its depth which is the maximum number of nodes that an example can pass through before obtaining its prediction. Intuitively, having no limit on the depth can lead to over-fit the data as one branch could be built to predict exactly the label of each training example. On the other hand, having a depth near 1 may produce a model that under-fits the data.

The goal when learning a tree is to find at each node the best feature and the best test on the value of this feature to split the training examples located in that node. For classification, this notion of "best" is quantified by a criterion called the Gini impurity:

$$GI(N) = \sum_{i=1}^{c} p_i(1 - p_i),$$

where $p_i$ is the proportion of examples of the $i$th class in the node $N$. In binary classification, $GI \in [0, 0.5]$ where $GI = 0$ indicates that there are examples of only one class in the node, and $GI = 0.5$ indicates that there are as many examples of both classes in the node. Decision tree algorithms [Breiman et al., 1984] aim to find a feature and a threshold that lead to a decrease of the Gini impurity (see Figure 1.8 for an example).

### 1.2.4 Neural networks

In Rosenblatt [1958], a simple neural network, called perceptron, is introduced as follows

$$h(\boldsymbol{x}) = g(\boldsymbol{w} \cdot \boldsymbol{x} + b),$$

where $g$ is called an activation function and the goal is to learn the vector $\boldsymbol{w}$ and the bias $b$. It is worth noting that it takes a similar form as that of a linear SVM where $g = \text{sign}$. A common type of neural networks is the multi-layer perceptron built upon the aggregation of several perceptrons organized in layers. For example, a basic 3-layer perceptron can be built as follows

$$h_3(\boldsymbol{x}) = g_3\left(\boldsymbol{w}_3 \cdot \begin{pmatrix} h_1(\boldsymbol{x}) = g_1(\boldsymbol{w}_1 \cdot \boldsymbol{x} + b_1) \\ h_2(\boldsymbol{x}) = g_2(\boldsymbol{w}_2 \cdot \boldsymbol{x} + b_2) \end{pmatrix} + b_3\right).$$

It has three layers *(i)* the input layer composed of $d$ neurons which is the number of features in the input space, *(ii)* a hidden layer composed of two neurons defined by the two perceptrons $h_1$ and $h_2$ and *(iii)* the output layer composed of one neuron defined by the last perceptron $h_3$.

Most neural networks can be seen as multi-layer perceptrons where the differences between them can be: the number of hidden layers, the number of neurons in each layer, the activation functions, the algorithm used to learn the parameters $\boldsymbol{w}$ and $b$ of each perceptron...

There has been during the last years a large attention to deep neural networks [Goodfellow et al., 2016] due to their excellent capability to learn from large set of examples. Current state-of-the art deep neural networks are especially good among others to address computer visions and natural language processing tasks.

An interesting characteristics of deep neural networks is that we can reuse a part of these networks to handle a different task. This is especially convenient as training these models can be very expensive in terms of computation time and hardware infrastructure. For example, the pioneering work of Krizhevsky et al. [2012] train a deep neural network using millions of images to find if an object among 1000 classes of object is present in an image. This pre-trained network can be re-used for a different task involving images, for example by using the network as a feature extractor where the features are produced as the output of one of its hidden layers. This strategy is exploited in Chapter 4 to quickly extract a set of features for images that are used to evaluate our proposed method.

### 1.2.5 Boosting

The idea behind boosting [Schapire, 1990] is to use a machine learning algorithm to build iteratively a set of models $h$ and then to return as final predictor an aggregation of these models. In binary classification, the final boosted model noted $H$ takes the form of the following weighted sum:

$$H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha^t h_{\mathbf{a}^t}(\boldsymbol{x})\right),$$

where $\alpha^t$ is the weight given to the $t^{\text{th}}$ model, and where $T$ is the number of models trained, *i.e.*, the number of iterations of the algorithm. The individual models are usually trained to be complementary to the others.

In the seminal work of Schapire and Singer [1999], the algorithm Adaboost resorts to the minimization of the exponential loss by gradient descent. At each iteration, the distribution of the training examples is updated, giving more weight to the examples that have been miss-classified by the previous classifiers. One advantage of Adaboost is that, under a weak assumption over the base classifiers, the final model comes with strong generalization guarantees.

In the gradient boosting framework proposed by Friedman [2001], at each iteration, a regression model is trained to predict the residuals of the examples defined as minus the partial derivative of any empirical loss with respect to the current ensemble. The goal is to correct step by step the errors made by the previous ensemble.

We show in Figure 1.9 the behavior of Adaboost [Schapire and Singer, 1999] and Gradient Boosting [Friedman, 2001] where we plug decision trees of depth 1 as base models. We can see at the first iteration that a single tree of depth 1 fails to separate well the data. However, when increasing the number of trees, we quickly obtain a model that correctly predicts all the training examples.

## 1.3 Methodological building blocks

We investigate in this thesis how to obtain relevant representations of the data in the difficult setting where only a few examples of interest are available to learn from. The goal is that a model trained with the new representation will present better performances than a model learned upon the original representation. In the following, we recall three frameworks used as building blocks in our contributions. Metric learning (see the two surveys: [Bellet et al., 2015, Kulis, 2013]), is the basis of Chapter 2 where we optimize a metric inducing a new representation space that tends to be more effective than state-of-the-art metrics in the presence of a class imbalance. Kernel random Fourier features [Rahimi and Recht, 2008] are used in Chapter 3 to learn new features helping to generalize with few labeled examples. Finally the optimal transportation theory [Villani, 2008] is exploited in Chapter 4 to reduce the discrepancy between the features of a target domain where no labels are available, and the features of a source labeled set.

### 1.3.1 Metric learning

Metric learning is a sub-field of representation learning that consists in designing a pairwise function able to capture the dis/similarity between two data points. This is a key issue in machine learning as such metrics are at the core of many algorithms, like $k$NN, SVMs...

Many metric learning algorithms are of the family of methods that construct a generalized version of the Mahalanobis distance [Mahalanobis, 1936] defined as

$$d_{\mathbf{M}}(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^\top \mathbf{M} (\boldsymbol{x} - \boldsymbol{x}')}$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is symmetric and positive semi-definite (PSD), *i.e.*, for all non-null vector $\boldsymbol{x} \in \mathbb{R}^d$ then

$$\boldsymbol{x}^\top \mathbf{M} \boldsymbol{x} \geq 0.$$

This distance allows to retrieve the Euclidean distance by setting $\mathbf{M}$ as the identity matrix of dimension $d$. Because $\mathbf{M}$ is PSD, another interesting property is that there exists a matrix $\mathbf{L} \in \mathbb{R}^{r \times d}$ where $r$ is the rank of $\mathbf{M}$ such that

$$\mathbf{M} = \mathbf{L}^\top \mathbf{L}.$$

Thus, the Mahalanobis distance between two points $\boldsymbol{x}$ and $\boldsymbol{x}'$ is equal to their Euclidean distance after having projected linearly $\boldsymbol{x}$ and $\boldsymbol{x}'$ in the $r$-dimensional space, *i.e.*,

$$\begin{aligned} d_{\mathbf{M}}(\boldsymbol{x}, \boldsymbol{x}') =& \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^\top \mathbf{M} (\boldsymbol{x} - \boldsymbol{x}')} \\ =& \sqrt{(\mathbf{L}\boldsymbol{x} - \mathbf{L}\boldsymbol{x}')^\top (\mathbf{L}\boldsymbol{x} - \mathbf{L}\boldsymbol{x}')} \,. \end{aligned}$$

The goal of metric learning algorithms is then to construct the matrix $\mathbf{M}$ or $\mathbf{L}$ that induces a distance measure suited for a given task. Most metric learning algorithms optimize a loss function which aims at bringing closer examples of the same label while pushing apart examples of different labels. In practice, metric learning is usually performed with pairwise constraints—two data points $\boldsymbol{x}$ and $\boldsymbol{x}'$ should be dis/similar [Davis et al., 2007, Lu et al., 2013, Weinberger and Saul, 2009, Xiang et al., 2008, Xing et al., 2003, Zadeh et al., 2016]—or relative constraints—a data point $\boldsymbol{x}$ should be more similar to another $\boldsymbol{x}'$ than to a third one $\boldsymbol{x}''$ [Lee et al., 2008, Schultz and Joachims, 2004, Weinberger and Saul, 2009, Zheng et al., 2011].

We will show in Chapter 2 that when learning with few labeled examples of a class but a large amount of another class, existing metric learning algorithms tend to favor the majority class. To face this issue, we will propose a new metric learning algorithm aiming to be as good on both classes of examples. We will exploit the uniform stability framework to derive guarantees on the learned metric $\mathbf{M}$.

### 1.3.2 Random Fourier features (RFF)

The RFF framework introduced by Rahimi and Recht [2008] allows to approximate kernel functions in order to speed up the learning of algorithms using such kernels. A kernel can be defined as a function

$$k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R},$$

that takes as input two examples and returns a measure of similarity between them. This framework allows to approximate the set of kernels that are shift-invariant, meaning by abuse of notation that

$$k(\boldsymbol{x}, \boldsymbol{x}') = k(0, \boldsymbol{x} - \boldsymbol{x}') = k(\boldsymbol{x} - \boldsymbol{x}') = k(\boldsymbol{\delta}) \quad \text{with} \quad \boldsymbol{\delta} = \boldsymbol{x} - \boldsymbol{x}'.$$

When a kernel is shift-invariant, it is possible to define a distribution $p(\boldsymbol{\omega})$ as the Fourier transform of the kernel [Rudin, 1962]:

$$p(\boldsymbol{\omega}) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} k(\boldsymbol{\delta}) e^{-i\boldsymbol{\omega}\cdot\boldsymbol{\delta}} d\boldsymbol{\delta}. \tag{1.3}$$

In this context, Rahimi and Recht [2008] show that the kernel can be rewritten as

$$\begin{aligned} k(\boldsymbol{x} - \boldsymbol{x}') =& \mathop{\mathbb{E}}_{\boldsymbol{\omega} \sim p} \cos\big(\boldsymbol{\omega} \cdot (\boldsymbol{x} - \boldsymbol{x}')\big) \\ \simeq& \frac{1}{K} \sum_{j=1}^{K} \cos(\boldsymbol{\omega}_j \cdot (\boldsymbol{x} - \boldsymbol{x}')), \end{aligned}$$

where the larger the number of random features $K$, the more accurate the resulting approximation of the kernel. Given $K$ vectors $\{\boldsymbol{\omega}_j\}_{j=1}^{K} \sim p^K$, they further propose to approximate the kernel as

$$k(\boldsymbol{x}, \boldsymbol{x}') \simeq z(\boldsymbol{x}) \cdot z(\boldsymbol{x}'),$$

where $z$ is a mapping in the random feature space defined as

$$z(\boldsymbol{x}) = \sqrt{\frac{1}{K}} \left[\cos(\boldsymbol{\omega}_1 \cdot \boldsymbol{x}), \ldots, \cos(\boldsymbol{\omega}_K \cdot \boldsymbol{x}), \sin(\boldsymbol{\omega}_1 \cdot \boldsymbol{x}), \ldots, \sin(\boldsymbol{\omega}_K \cdot \boldsymbol{x})\right].$$

The interest of this framework is to project the points in the random feature space, and then to train a linear model in this space. Doing so, we benefit both from the non-linearity induced by the kernel approximated, and the fast training time of the linear model. However, if the considered kernel is not suited for the task at hand, its approximated version will not be more suited because it only tends to give the same value as the kernel. In this sense, the RFF method only allows to speed up the learning time of a kernel algorithm without improving its effectiveness to handle a task. To obtain a new kernel suited for a given task, several works have extended this technique by allowing one to adapt the RFF approximation directly from the labeled training data [Agrawal et al., 2019, Letarte et al., 2019, Sinha and Duchi, 2016]. This is also the focus of our contribution in Chapter 3 where we build at the same time a representation of the data with RFF and a classification model with gradient boosting to obtain a model that generalizes well in the presence of few labeled examples.

### 1.3.3 Optimal transport

The theory of optimal transport has been introduced by Monge [1781] and was recently revisited by Villani [2008]. In essence, this theory gives a mathematically founded tool that allows to align arbitrary probability distributions in an optimal way.

In the discrete case, it can be formalized as follows. Let $\widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{S} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{\delta}_{\boldsymbol{x}_i^S}$ and $\widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{T} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\delta}_{\boldsymbol{x}_i^T}$ be two empirical probability measures defined as uniformly weighted sums of Diracs with mass at locations defined on two sets $S = (\boldsymbol{X}^S, \boldsymbol{Y}^S)$ with $\boldsymbol{X}^S \in \mathbb{R}^{m \times d}$ and $T = (\boldsymbol{X}^T, \boldsymbol{Y}^T)$ with $\boldsymbol{X}^T \in \mathbb{R}^{n \times d}$ drawn according to arbitrary probability distributions $\mathcal{D}^S$ and $\mathcal{D}^T$. The Monge-Kantorovich problem consists in finding a probabilistic coupling $\boldsymbol{\gamma}$ defined as a joint probability distribution over $\boldsymbol{X}^S \times \boldsymbol{X}^T$ that minimizes the cost of transport *w.r.t.* a metric $c : \mathcal{X}^S \times \mathcal{X}^T \to \mathbb{R}_+$:

$$\boldsymbol{\gamma}^* = \underset{\boldsymbol{\gamma} \in \Pi(\widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{S}, \widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{T})}{\arg\min} \langle \boldsymbol{\gamma}, \boldsymbol{C} \rangle_F, \tag{1.4}$$

where $\langle\cdot,\cdot\rangle_F$ is the Frobenius dot product, $\Pi(\widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{S}, \widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{T}) = \{\boldsymbol{\gamma} \in \mathbb{R}_+^{m \times n} | \boldsymbol{\gamma}\mathbf{1} = \widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{S}, \boldsymbol{\gamma}^\top\mathbf{1} = \widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{T}\}$ is a set of doubly stochastic matrices and $\boldsymbol{C}$ is a dissimilarity matrix, *i.e.*, for $\boldsymbol{x}_i^S \in \boldsymbol{X}^S$ and $\boldsymbol{x}_j^T \in \boldsymbol{X}^T$, we have $C_{ij} = c(\boldsymbol{x}_i^S, \boldsymbol{x}_j^T)$ which defines the energy needed to move a probability mass from $\boldsymbol{x}_i^S$ to $\boldsymbol{x}_j^T$. This problem admits one or several optimal solutions $\boldsymbol{\gamma}^*$ and defines a metric on the space of probability measures (called the Wasserstein distance) as follows:

$$W(\widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{S}, \widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{T}) = \min_{\boldsymbol{\gamma} \in \Pi(\widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{S}, \widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{T})} \langle \boldsymbol{\gamma}, \boldsymbol{C} \rangle_F. \tag{1.5}$$

We supply an example showing the computation of $\boldsymbol{\gamma}^*$ on a toy example in Figure 1.10.

Despite its elegance and simplicity, the formulation of optimal transport given in Equation (1.4) (abbreviated **OT**) is a Linear Programming problem that does not scale well because of its computational complexity.

In order to tackle this issue, Cuturi [2013] proposed to add the entropic regularization of $\boldsymbol{\gamma}$ to the Equation (1.4) leading to the following optimization problem:

$$\boldsymbol{\gamma}^* = \operatorname*{arg\,min}_{\boldsymbol{\gamma} \in \Pi(\widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{S}, \widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^{T})} \langle \boldsymbol{\gamma}, \boldsymbol{C} \rangle_F - \frac{1}{\lambda} E(\boldsymbol{\gamma}), \tag{1.6}$$

where $E(\boldsymbol{\gamma}) = -\sum_{ij} \gamma_{ij} \log \gamma_{ij}$. The regularized optimal transport (abbreviated **OT2**) allows the source instances to be transported more or less uniformly to the target instances based on a hyper-parameter $\lambda$ and can be optimized efficiently with the linear time Sinkhorn-Knopp algorithm [Knight, 2008].

Based on the solutions given by Equations (1.4) and (1.6), we propose in Chapter 4 in the unsupervised domain adaptation context where no labeled examples are available in a target domain, to measure how similar the features of a source domain with labeled examples are to the features of the target domain. These similarities can then be used to build a better classification model on the target domain by discarding the most dissimilar features between the two domains.

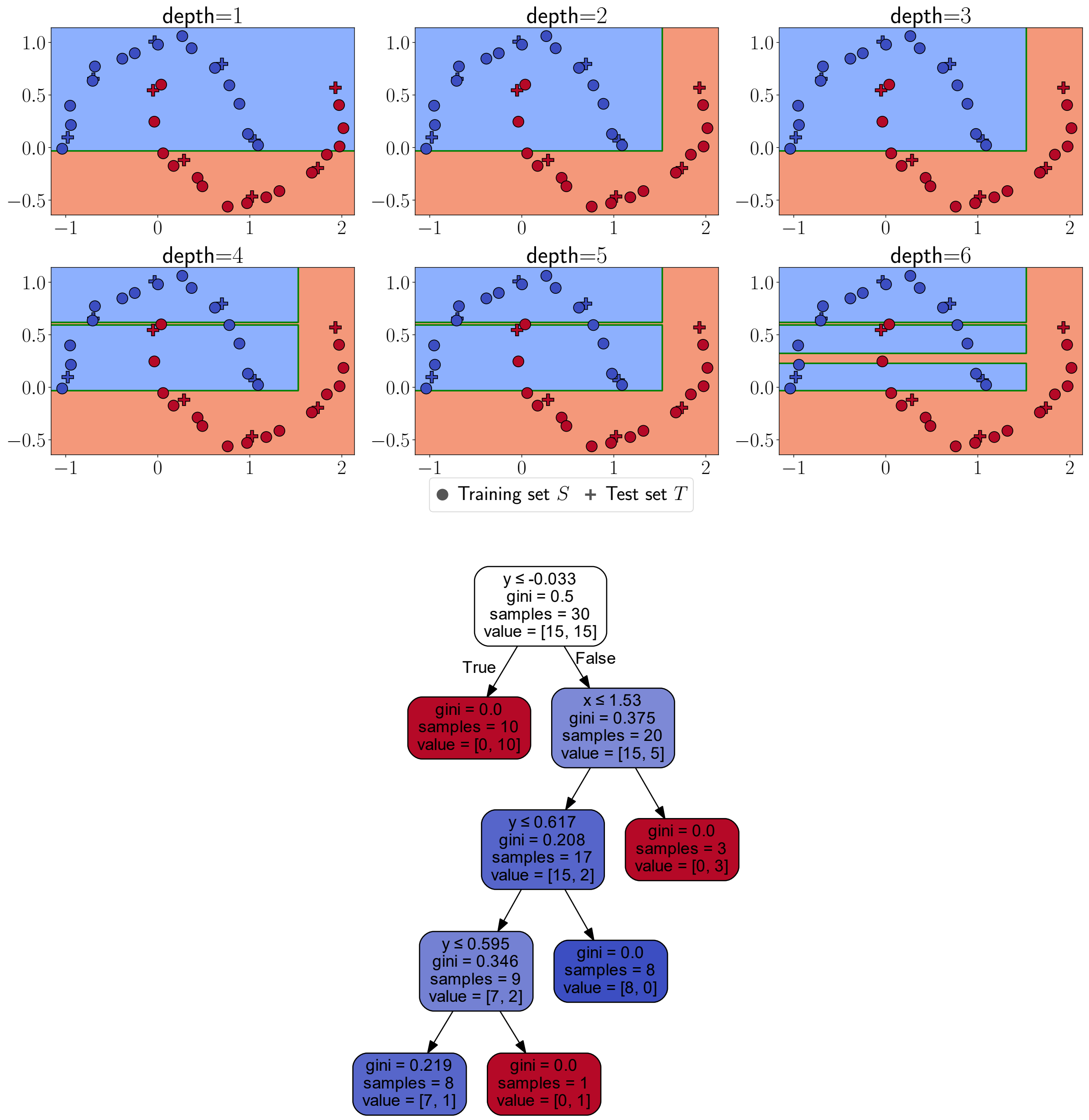


*Figure 1.8: Partition of the space different depths of the decision tree (top) and the decision tree with depth 4 (bottom). Larger depths lead to models that better fit the training set, but that may behave poorly on the test examples.*

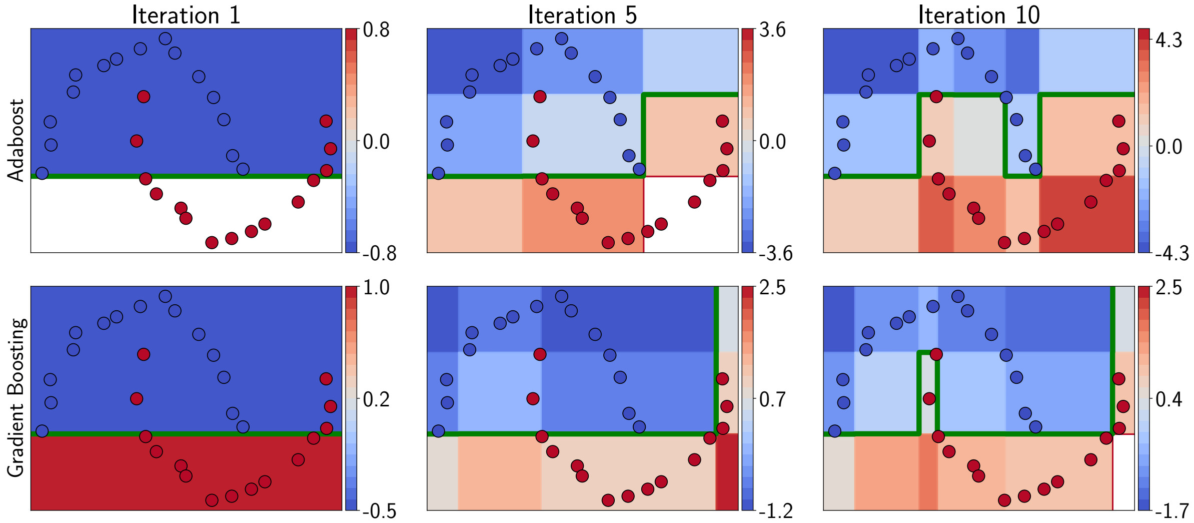
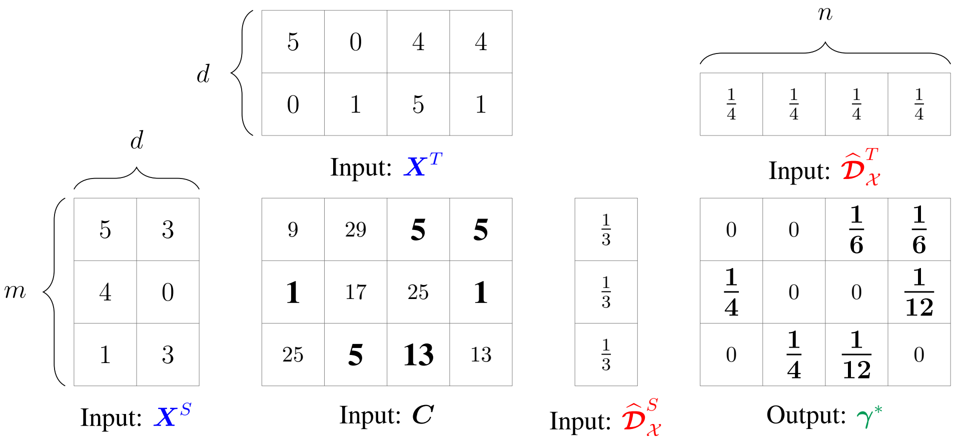


*Figure 1.9: Adaboost model [Schapire and Singer, 1999] and Gradient Boosting model [Friedman, 2001] with decision trees of depth* 1 *as base learner using* 1*,* 5 *and* 10 *iterations. Note that the white areas are where the predictions are equal to* 0 *which are considered a positive value (i.e., the red class).*



$$d\left\{\begin{array}{|c|c|c|c|}\hline 5 & 0 & 4 & 4 \\ \hline 0 & 1 & 5 & 1 \\ \hline\end{array}\right.$$

Input: $\boldsymbol{X}^T$

$$\overbrace{\begin{array}{|c|c|c|c|}\hline \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \hline\end{array}}^{n}$$

Input: $\widehat{\mathcal{D}}_{\mathcal{X}}^{T}$

$$m\left\{\overbrace{\begin{array}{|c|c|}\hline 5 & 3 \\ \hline 4 & 0 \\ \hline 1 & 3 \\ \hline\end{array}}^{d}\right.$$

Input: $\boldsymbol{X}^S$

$$\begin{array}{|c|c|c|c|}\hline 9 & 29 & \mathbf{5} & \mathbf{5} \\ \hline \mathbf{1} & 17 & 25 & \mathbf{1} \\ \hline 25 & \mathbf{5} & \mathbf{13} & 13 \\ \hline\end{array}$$

Input: $\boldsymbol{C}$

$$\begin{array}{|c|}\hline \frac{1}{3} \\ \hline \frac{1}{3} \\ \hline \frac{1}{3} \\ \hline\end{array}$$

Input: $\widehat{\mathcal{D}}_{\mathcal{X}}^{S}$

$$\begin{array}{|c|c|c|c|}\hline 0 & 0 & \mathbf{\frac{1}{6}} & \mathbf{\frac{1}{6}} \\ \hline \mathbf{\frac{1}{4}} & 0 & 0 & \mathbf{\frac{1}{12}} \\ \hline 0 & \mathbf{\frac{1}{4}} & \mathbf{\frac{1}{12}} & 0 \\ \hline\end{array}$$

Output: $\gamma^*$

*Figure 1.10: Example of an optimal transport problem between two sets of examples* $\boldsymbol{X}^S$ *and* $\boldsymbol{X}^T$ *where* $\boldsymbol{C}$ *is the squared Euclidean distance and both* $\widehat{\mathcal{D}}_{\mathcal{X}}^{S}$ *and* $\widehat{\mathcal{D}}_{\mathcal{X}}^{T}$ *are uniform empirical probability measures.*

# Chapter 2

# Metric Learning from Imbalanced Data with Generalization Guarantees

**This chapter is based on the following publications**

Léo Gautheron, Emilie Morvant, Amaury Habrard and Marc Sebban. Metric Learning from Imbalanced Data with Generalization Guarantees. In *Pattern Recognition Letters*, volume 133, pages 298-304. 2020 [Gautheron et al., 2020c].

Léo Gautheron, Amaury Habrard, Emilie Morvant and Marc Sebban. Metric Learning from Imbalanced Data. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019, Portland, United States [Gautheron et al., 2019b].

Léo Gautheron, Amaury Habrard, Emilie Morvant and Marc Sebban. Apprentissage de métrique pour la classification supervisée de données déséquilibrées. In *Conférence sur l'Apprentissage automatique (CAp)*, 2018, Rouen, France [Gautheron et al., 2018a].

**Abstract**

Since many machine learning algorithms require a distance metric to capture dis/similarities between data points, *metric learning* has received much attention during the past two decades. Surprisingly, very few methods have focused on learning a metric in an imbalanced scenario where the number of positive examples is much smaller than the negatives, and even fewer derived theoretical guarantees in this setting. Here, we address this difficult task and design a new Mahalanobis metric learning algorithm (**IML**) which deals with class imbalance. We further prove a generalization bound involving the proportion of positive examples using the uniform stability framework. The empirical study performed on a wide range of datasets shows the efficiency of **IML**.

## 2.1 Introduction and related work

In this chapter, we focus on the family of metric learning algorithms (see the two surveys: [Bellet et al., 2015, Kulis, 2013]) that construct a generalized version of the Mahalanobis distance in the presence of a class of rare examples and another class containing most of the training data.

Two famous representatives of Mahalanobis distance learning are **LMNN** (Large Margin Nearest Neighbor [Weinberger and Saul, 2009]) and **ITML** (Information-Theoretic Metric Learning [Davis et al., 2007]), which are both designed to improve the accuracy in the latent space of the $k$NN classification rule recalled in Section 1.2.1.

The idea behind **LMNN** is to learn the matrix $\mathbf{M}$ that parametrizes the Mahalanobis distance and satisfies a set of similar constraints $\mathcal{S}$, and a set of relative constraints $\mathcal{R}$, defined as

$$\begin{aligned} \mathcal{S} &= \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \mid y_i = y_j \text{ and } \boldsymbol{x}_j \text{ belongs to the k neighbors of } \boldsymbol{x}_i\}, \\ \text{and } \mathcal{R} &= \{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) \mid (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{S} \text{ and } y_i \neq y_k\}. \end{aligned}$$

Then, the metric is learned by optimizing the following problem:

$$\begin{aligned} \underset{\mathbf{M}\succeq 0}{\arg\min} \quad & (1-\mu) \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathcal{S}} d^2_{\mathbf{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \mu \sum_{i,j,k} \xi_{ijk} \\ \text{s.t. } & d^2_{\mathbf{M}}(\boldsymbol{x}_i, \boldsymbol{x}_k) - d^2_{\mathbf{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 1 - \xi_{ijk} \quad \forall (\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) \in \mathcal{R}. \end{aligned}$$

where $\mathbf{M} \succeq 0$ denotes the constraint that $\mathbf{M}$ should be PSD, as introduced in Section 1.3.1. The idea of this method is to optimize a trade-off (controlled by $\mu \in [0, 1]$) between a minimization of the distance between similar pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and under the constraint that any example $\boldsymbol{x}_k$ with a different label is farther from $\boldsymbol{x}_i$ than from $\boldsymbol{x}_j$. In practice, these constraints are relaxed through the use of slack variables $\xi_{ijk}$.

The **ITML** algorithm also considers two randomly selected sets of similar and dissimilar pairs noted $\mathcal{S}$ and $\mathcal{D}$, and optimizes $\mathbf{M}$ by minimizing the following problem:

$$\begin{aligned} \underset{\mathbf{M}\succeq 0}{\arg\min} \quad & D_{ld}(\mathbf{M}, \mathbf{M}_0) \\ \text{s.t. } & d^2_{\mathbf{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq u \quad \forall (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{S} \\ & d^2_{\mathbf{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq v \quad \forall (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{D}, \end{aligned}$$

where $u$ and $v$ are safety margin parameters. The goal is to learn a matrix $\mathbf{M}$ sufficiently close to a prior matrix $\mathbf{M}_0$ under the LogDet divergence $D_{ld}$ and the constraints that the examples in $\mathcal{S}$ must be closer than a value $u$ and that the instances in $\mathcal{D}$ must be significantly far away with a distance larger than $v$ where $v > u$.

Without being exhaustive, another kind of Mahalanobis distance learning algorithm is given with **GMML** (Geometric Mean Metric Learning) [Zadeh et al., 2016] that also considers two

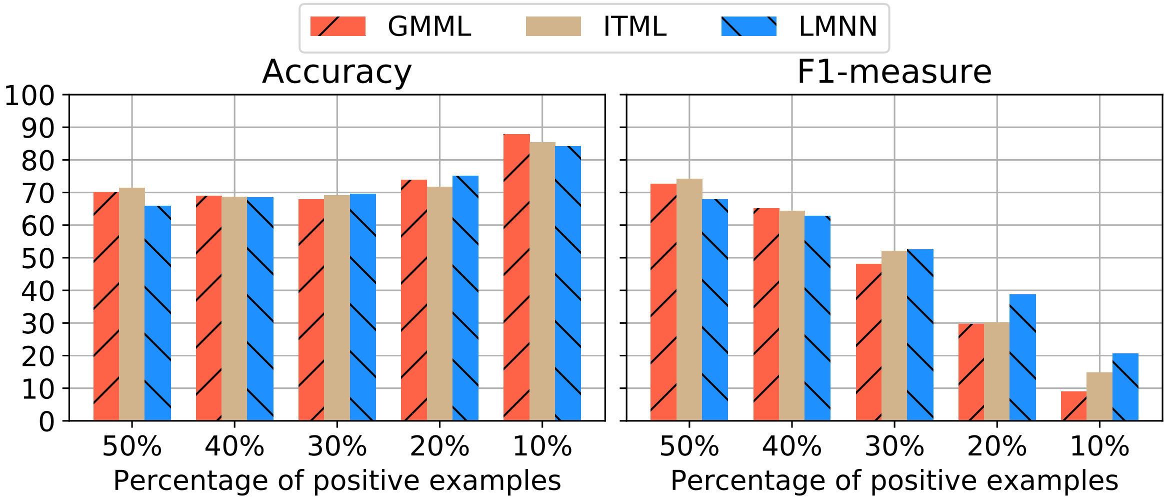


*Figure 2.1: Illustration on the* SPECTFHEART *dataset of the negative impact of classic metric learning algorithms when facing an increasing imbalance in the dataset. On the left, as the proportion of minority examples decreases (the positive class), the* 3*NN algorithm with the learned metrics tends to classify all the examples as members of the majority class, with an accuracy close to* 100%*. On the right, using the F1-measure (see its definition given in Section 1.1.2) , we see that the learned metrics plugged in a kNN actually miss many positives.*

random sets of pairs $\mathcal{S}$ and $\mathcal{D}$ and whose objective function is:

$$\arg\min_{\mathbf{M}\succeq 0} \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathcal{S}} d_{\mathbf{M}}(\boldsymbol{x}_i,\boldsymbol{x}_j) + \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathcal{D}} d_{\mathbf{M}^{-1}}(\boldsymbol{x}_i,\boldsymbol{x}_j).$$

To find the matrix $\mathbf{M}$, they consider the two following matrices:

$$\boldsymbol{S} = \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathcal{S}} (\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top$$

$$\boldsymbol{D} = \sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j)\in\mathcal{D}} (\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top.$$

The goal is to find $\mathbf{M}$, inside the set of PSD matrices, along the geodesic curve between $\boldsymbol{D}$ and $\boldsymbol{S}^{-1}$ parameterized by $t \in [0,1]$ that controls if it is closer to $\boldsymbol{D}$ or $\boldsymbol{S}^{-1}$. The main advantage of this method is that $\mathbf{M}$ can be computed explicitly from $\boldsymbol{S}^{-1}$ and $\boldsymbol{D}$ and does not require a costly optimization process as for **LMNN** and **ITML**.

In light of these learning procedures, it is worth noticing that the loss functions optimized in **LMNN**, **ITML** and **GMML** (and in most pairwise metric learning methods) tend to favor the majority class as there is no distinction between the constraints involving examples of the majority class and the constraints on the minority class. This strategy is thus not well suited when dealing with imbalanced datasets. An illustration of this phenomenon on the SPECTFHEART dataset from the UCI[1] repository is shown in Figure 2.1. We observe that decreasing the proportion of minority examples tends to generate a metric which classifies (with

[1] `https://archive.ics.uci.edu/ml/datasets.html`

a $k$NN rule) all the examples as the majority class, thus leading to an accuracy close to 100%. On the other hand, the F1-measure [Van Rijsbergen, 1974], commonly[2] used in imbalanced settings [Chandola et al., 2009, López et al., 2013], decreases with the proportion of positives, showing that the classifier missed many positives, usually considered as the examples of interest.

This problem of learning from imbalanced data has been widely tackled in the literature [Branco et al., 2016, He and Garcia, 2009]. Classic methods typically make use of over/under-sampling techniques [Drummond and Holte, 2003, Estabrooks et al., 2004, Liu et al., 2008, Aggarwal, 2013] or create synthetic examples in the neighborhood of the minority class—*e.g.,* using SMOTE-like strategies [Chawla et al., 2002, 2003, Han et al., 2005] or resorting to adversarial techniques [Douzas and Bacao, 2018]. However, these methods may lead to over or under-fitting and are often subject to an inability to generate enough diversity, especially in a highly imbalanced scenario. Other strategies aim at addressing imbalanced situations directly during the learning process. They include cost-sensitive methods [Elkan, 2001, Zadrozny et al., 2003] which require prior knowledge on the miss-classification costs, the optimization of imbalance-aware criteria [Frery et al., 2017, McFee and Lanckriet, 2010, Vogel et al., 2018] which are often non convex, or ensemble methods based on bagging and boosting strategies [Galar et al., 2011] that can be computationally expensive.

Unlike the state of the art, we suggest in this chapter to address the problem of learning from imbalanced data by optimizing a metric suited to scenarios where the positive data are very scarce. As far as we know, very few methods were designed in this setting. Feng et al. [2018] propose to regularize a standard metric learning problem by using the KL-divergence between the classes. Wang et al. [2018] propose **IMLS** that learns a classic metric and then performs a sampling on the training data to account the imbalance. However, as we will see in our experimental study, better performances can be achieved by resorting to a metric dedicated specifically to deal with the imbalance of the application at hand. Deep metric learning methods have also received attention by the community to address the problem of imbalanced data [Liu et al., 2019, Wang et al., 2019]. However these methods often require large training datasets, like in visual tasks, a requirement which is not always fulfilled by the application at hand. Moreover, it is worth noticing that none of the previous approaches come with theoretical guarantees, a gap we will fill in this chapter. In order to implicitly control the rates of false positives and false negatives, we propose a new algorithm, called **IML** for Imbalanced Metric Learning, which accounts carefully the nature of the pairwise constraints (by decomposing them with respect to the labels involved in the pairs) and weights their impact in the loss function so as to account the imbalance. Beyond this algorithmic contribution, we further provide a theoretical analysis of **IML** using the uniform stability framework [Bousquet and Elisseeff, 2002] presented in Section 1.1.5. We derive the first generalization bound which has the advantage to involve the proportion of minority examples. This bound provides some insight into the way to tune

[2] As indicated in Section 1.1.2, the F1-measure is much more adapted to imbalanced scenarios since it does not involve the true negatives but considers both the false positives and the false negatives.

the weighting parameters to counterbalance the negative impact of imbalanced datasets.

**Organization of the chapter.** Section 2.2 introduces the notations and the principle of classical Mahalanobis metric learning. Section 2.3 describes our algorithm **IML** which takes the form of a simple regularized convex problem. Section 2.4 is dedicated to the theoretical analysis. We perform an experimental study of our approach in Section 2.5 before concluding in Section 2.6.

## 2.2 Notations and setting

In this chapter, we deal with binary classification tasks and follow the same notations as in Chapter 1. We assume that the training set is defined as $S = S^+ \cup S^-$, with $S^+$ the set of positive examples and $S^-$ the set of negative examples such that the number of positives $m^+ = |S^+|$ is smaller than the number of negatives $m^- = |S^-|$ (we say that $+1$ is the minority class and $-1$ the majority one). We aim at constructing a Mahalanobis distance which induces a new space in which a $k$NN classifier will work well on both classes.

Mahalanobis metric learning algorithms [Bellet et al., 2015, Cao et al., 2016, Jin et al., 2009] can usually be expressed as follows:

$$\min_{\mathbf{M}\succeq 0} F(\mathbf{M}) = \frac{1}{m^2} \sum_{(\mathbf{z},\mathbf{z}')\in S^2} \ell(\mathbf{M},\mathbf{z},\mathbf{z}') + \lambda Reg(\mathbf{M}), \tag{2.1}$$

where one wants to minimize the trade-off between a convex loss $\ell$ over all pairs of examples and a regularization $Reg$ under the PSD constraint $\mathbf{M} \succeq 0$.

The major drawback of this classical formulation is that the loss gives the same importance to any pair of labeled examples $(\mathbf{z},\mathbf{z}')$ whatever the labels $y$ and $y'$. Intuitively, this is not well suited to imbalanced scenarios where one wants to focus more on the minority class (think, for example, about anomaly detection [Chandola et al., 2009]). Some metric learning algorithms [Weinberger and Saul, 2009, Zadeh et al., 2016] allow to weight the role played by the similar and dissimilar/relative constraints, but they do not directly take into account the labels of the examples.

To tackle these drawbacks, we propose in the next section **IML**, a metric learning algorithm able to deal with imbalanced data.

## 2.3 IML: Imbalanced Metric Learning

Our algorithm is built on the simple idea consisting in decomposing further the sets of similar and dissimilar constraints based on the two labels involved in the constraints. Each set can then be weighted differently during the optimization to reduce the negative effect of the imbalance.

Starting from Equation (2.1), let us decompose the loss function $\ell$; we have for all $(\mathbf{z},\mathbf{z}') \in \mathcal{Z}^2$

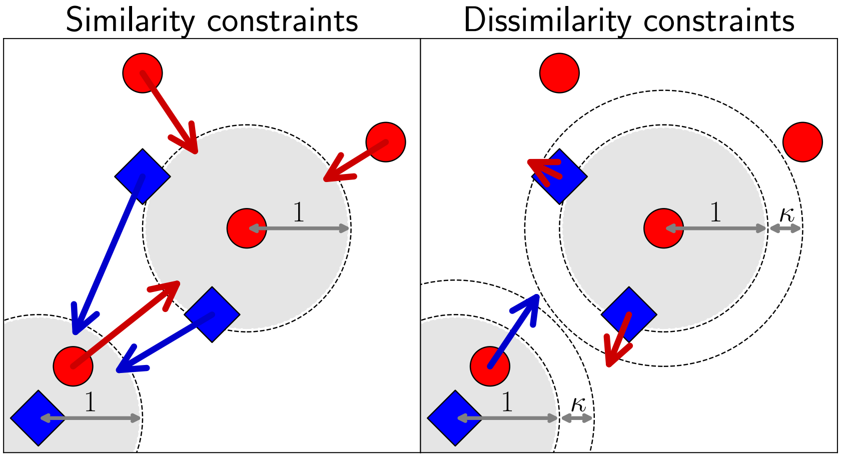


*Figure 2.2: Illustration of the behavior of our loss $\ell$ defined in Equation* (2.2)*. On the left, the similarity constraints (loss $\ell_1$) aim at bringing examples of the same class at a distance less than* 1*. On the right, the dissimilarity constraints (loss $\ell_2$) aim at pushing away examples of different classes at a distance larger than* 1 *plus a margin $\kappa$.*

and for all $\mathbf{M} \in \mathbb{R}^{d\times d}$ :

$$\ell(\mathbf{M},\mathbf{z},\mathbf{z}') = \begin{cases} a\ell_1(\mathbf{M},\mathbf{z},\mathbf{z}') & \text{if } y=+1 \text{ and } y'=+1,\\ (1{-}a)\ell_1(\mathbf{M},\mathbf{z},\mathbf{z}') & \text{if } y=-1 \text{ and } y'=-1,\\ b\ell_2(\mathbf{M},\mathbf{z},\mathbf{z}') & \text{if } y=+1 \text{ and } y'=-1,\\ (1{-}b)\ell_2(\mathbf{M},\mathbf{z},\mathbf{z}') & \text{if } y=-1 \text{ and } y'=+1, \end{cases} \tag{2.2}$$

with the two functions $\ell_1$ and $\ell_2$ defined as $\ell_1(\mathbf{M},\mathbf{z},\mathbf{z}') = [d^2_{\mathbf{M}}(\boldsymbol{x},\boldsymbol{x}') - 1]_+$ and $\ell_2(\mathbf{M},\mathbf{z},\mathbf{z}') = [1{+}\kappa{-}d^2_{\mathbf{M}}(\boldsymbol{x},\boldsymbol{x}')]_+$ and where $\kappa \geq 0$ is a margin parameter.

We illustrate in Figure 2.2 the behavior of the two sub-losses $\ell_1$ and $\ell_2$. The idea of $\ell_1$ is to bring examples of the same class at a distance less than 1 while $\ell_2$ aims to push far away examples of different classes at a distance larger than 1 plus a margin $\kappa$.

Both hyper-parameters $a$ and $b$ take values in $[0,1]$. The parameter $a$ controls the trade-off between bringing closer the minority examples and bringing closer the majority examples. While the second parameter $b$ controls the trade-off between keeping far away the majority examples from the neighborhood of minority ones, and keeping far away minority examples from the neighborhood of majority ones.

In addition to inserting Equation (2.2) into Equation (2.1), we need to set the regularization term $Reg(\mathbf{M})$. In order to avoid over-fitting, we propose to enforce $\mathbf{M}$ to be close to the identity matrix $\mathbf{I}$ such as $Reg(\mathbf{M}) = \|\mathbf{M}-\mathbf{I}\|^2_F$, with $\|\cdot\|_F$ the Frobenius norm. In other words, we aim at learning a Mahalanobis metric which is close to the Euclidean distance while satisfying the best the semantic constraints.

All things considered, our **IML** algorithm takes the form of the following convex problem:

$$\min_{\mathbf{M}\succeq 0} F(\mathbf{M}) = \frac{1}{m^2}\Bigg(\sum_{(\mathbf{z},\mathbf{z}')\in Sim^+} a\ell_1(\mathbf{M},\mathbf{z},\mathbf{z}') + \sum_{(\mathbf{z},\mathbf{z}')\in Sim^-} (1{-}a)\ell_1(\mathbf{M},\mathbf{z},\mathbf{z}') + \sum_{(\mathbf{z},\mathbf{z}')\in Dis^+} b\ell_2(\mathbf{M},\mathbf{z},\mathbf{z}') + \sum_{(\mathbf{z},\mathbf{z}')\in Dis^-} (1{-}b)\ell_2(\mathbf{M},\mathbf{z},\mathbf{z}')\Bigg) + \lambda\|\mathbf{M}-\mathbf{I}\|_F^2, \tag{2.3}$$

where the four sets $Sim^+$, $Dis^+$, $Dis^-$ and $Sim^-$ are defined as subsets of $S \times S$ respectively as: $Sim^+ \subseteq S^+ \times S^+$, $Dis^+ \subseteq S^+ \times S^-$, $Dis^- \subseteq S^- \times S^+$ and $Sim^- \subseteq S^- \times S^-$.

If we look more closely at the proposed Equation (2.3), when all pairs from $S \times S$ are involved, $Sim^+$ and $Sim^-$ contain respectively $m^+m^+$ and $m^-m^-$ pairs while $Dis^+$ and $Dis^-$ are composed respectively of $m^+m^-$ and $m^-m^+$ pairs. This means that the pairs in $Dis^+$ and $Dis^-$ are symmetric and these two sets might be merged. However, metric learning rarely considers all the possible pairs as it becomes quite inefficient in the presence of a large number of examples. Possible strategies to select the pairs include a random selection [Davis et al., 2007, Xiang et al., 2008, Xing et al., 2003, Zadeh et al., 2016] or a selection based on the nearest neighbors rule [Lu et al., 2013, Weinberger and Saul, 2009]. For this reason, it might make sense to separate the two sets $Dis^+$ and $Dis^-$ and allows to weight them differently as *(i)* they may not consider the same subsets of pairs, and *(ii)* may not capture the same geometric information. Another interpretation of such a decomposition in an imbalanced learning setting is the following: if $\mathbf{z}'$ is selected as belonging to the neighborhood of $\mathbf{z}$, the minimization of the four terms of Equation (2.3) can be seen as a nice way to implicitly optimize with a $k$NN rule the true positive, false negative, false positive and true negative rates respectively.

Among the two strategies to select the pairs, the selection based on the nearest neighbors is more adapted to an imbalanced scenario as it considers $k$ pairs for each training example from both the majority and minority classes. On the other hand, the random strategy just picks at random two examples to create a pair. Then with imbalanced data, it might be possible not to have any similar pair between two minority examples, thus focusing on the majority class. We will see experimentally in Section 2.5 that, as expected, the selection of the pairs based on the nearest neighbors rule performs better.

The fundamental difference between our formulation and classic metric learning formulations is that we separate in our loss the set of similar pairs $\mathcal{S}$ into two sets $Sim^+$ and $Sim^-$, and the set of dissimilar pairs $\mathcal{D}$ into $Dis^+$ and $Dis^-$. In a classic metric learning formulation, these four sets are all treated equally by giving them a weight of $\frac{1}{m^2}$. However in the presence of imbalanced data, the number of pairs in $Sim^+$ and $Dis^+$ which is in $O(m^+)$ is much smaller than in the sets $Sim^-$ and $Dis^-$ where the number of pairs is in $O(m^-)$. Intuitively, in the presence of imbalanced data, the terms in $O(m^+)$ will have a smaller impact on the loss function, thus, we aim at re-weighting these four sets to account the imbalance. We adopt a simple strategy consisting in giving a weight to each set that depends on its number of elements. We choose to give to the two sets $Sim^+$ and $Dis^+$ a weight $a = b = \frac{m^-}{m}$ and to the two sets $Sim^-$ and $Dis^-$ a weight $\frac{m^+}{m}$. This strategy allows us to give the same importance to the four terms in the loss function, no matter how imbalanced the data is. We will see experimentally that using this re-weighting instead of the weight $\frac{1}{m^2}$ greatly increases the performances when facing increasingly imbalanced data.

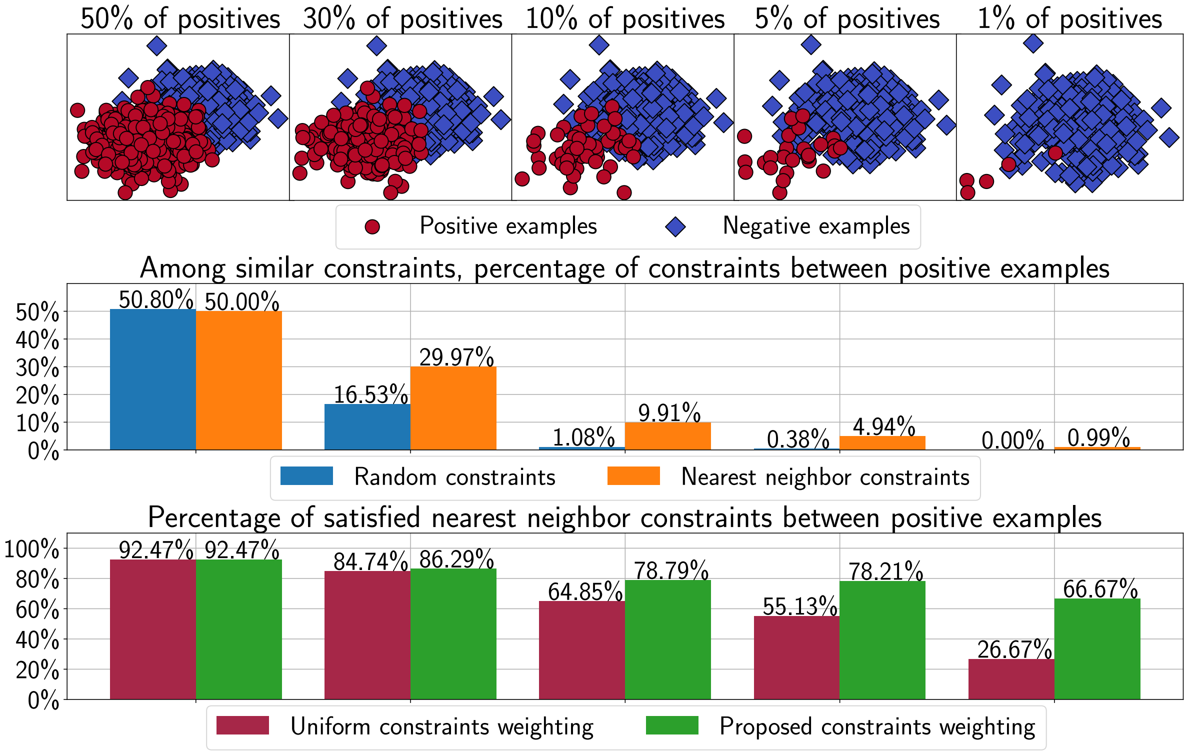


*Figure 2.3: Description of the two strategies of our* **IML** *algorithm to deal with the imbalance compared to classical metric learning methods. The top row of plots shows a toy dataset having an increasing class imbalance. The first strategy in the second row of plots consists in selecting constraints for each example rather than a global random selection to avoid having no similar constraints between minority examples in highly imbalanced scenarios. The second strategy (the third row of plots) consists in weighting the importance of the pairs depending on their labels to prevent the less represented constraints from being ignored during the optimization process and, as a result, not satisfied in the newly learned representation space.*

We illustrate in Figure 2.3 the two main strategies used by our proposed method to deal with the class imbalance. It depicts a toy dataset where two classes of points are represented initially in equal proportions, and where the percentage of examples of one of the two classes is gradually decreased from 50% to 1%. In the second row of plots, when increasing the imbalance, a random selection of the pairs of examples tends to select 0% of the pairs between minority examples when the data is extremely imbalanced with only 1% of minority examples. However, our first strategy to select the pairs of examples based on the $k$ nearest neighbor rule allows to obtain by construction the same percentage of pairs as the percentage of minority examples, because $k$ pairs are selected for every example.

We also notice in the third row of plots that the uniform weight given to the loss of every constraint by classical metric learning algorithms as done in Equation (2.1) is not adapted for imbalanced data. Indeed, the metric learning algorithm tends more and more to ignore the less represented constraints because they tend to not be satisfied in the new representation space (only 26.67% of constraints satisfied from 1% of positives). Our proposed weighting scheme allows to much better fulfill the similarity constraints between minority examples by re-balancing the importance given to the two classes in the objective function.

## 2.4 Generalization bound for IML

In this section, we provide a theoretical analysis of our algorithm using the uniform stability framework [Bousquet and Elisseeff, 2002] recalled in Section 1.1.5. This framework can be adapted to any metric learning algorithm [Bellet et al., 2015, Jin et al., 2009] taking the following form:

$$\min_{\mathbf{M}\succeq 0} G(\mathbf{M}) = \underbrace{\sum_{(\mathbf{z},\mathbf{z}')\in S^2} \ell(\mathbf{M},\mathbf{z},\mathbf{z}')}_{\widehat{R}(\mathbf{M})} + \lambda Reg(\mathbf{M})\,, \tag{2.4}$$

where $\widehat{R}(\mathbf{M})$ is the empirical loss of $\mathbf{M}$ on $S$, and $\ell$ is any loss function that is $q$-Lipschitz and $(\sigma, p)$-admissible as defined in the following.

**Definition 2** ($q$-Lipschitz function). *A function $f$ is $q$-Lipschitz w.r.t. its first argument if for any $u, v$,*

$$|f(u) - f(v)| \leq q|u - v|.$$

**Definition 3** (($\sigma, p$)-admissible function). *A loss $\ell$ is $(\sigma, p)$-admissible w.r.t. its first argument $\mathbf{M}$ if it is convex in $\mathbf{M}$ and if $\forall \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4$,*

$$|\ell(\mathbf{M},\mathbf{z}_1,\mathbf{z}_2) - \ell(\mathbf{M},\mathbf{z}_3,\mathbf{z}_4)| \leq \sigma|y_{12} - y_{34}| + p$$

*with $y_{ij} = +1$ if $y_i = y_j$ and $y_{ij} = -1$ otherwise.*

To prove a uniform stability-based generalization bound, the algorithm has to be stable— meaning that its output does not change significantly under a small modification of $S$— according to the following definition.

**Definition 4** ([Jin et al., 2009] Eq. (5)). *A metric learning algorithm has uniform stability in $\beta \geq 0$ w.r.t. the loss function $\ell$ if $\forall i \in \{1, \ldots, m\}$ the following holds*

$$\forall S \in \mathcal{Z}^m, \ \sup_{\mathbf{z},\mathbf{z}'} \left|\ell(\mathbf{M},\mathbf{z},\mathbf{z}') - \ell(\mathbf{M}^i,\mathbf{z},\mathbf{z}')\right| \leq \beta\,,$$

*where $\mathbf{M}$ is learned from $S$, and $\mathbf{M}^i$ is learned from $S^i$, the set obtained by replacing the $i^{th}$ example in $S$ by another also i.i.d. from $\mathcal{D}$.*

If an algorithm has uniform stability, then it is possible to derive an upper bound on its generalization error using the McDiarmid inequality [Bousquet and Elisseeff, 2002] recalled below.

**Theorem 2** (McDiarmid Inequality, [Bousquet and Elisseeff, 2002] Th. 2). *Let $G : \mathcal{Z}^m \to \mathbb{R}$ be any function for which there exists constants $c_i$, $\forall i \in \{1, \dots, m\}$ such that*

$$\sup_{S\in\mathcal{Z}^m, \mathbf{z}_i\in\mathcal{Z}} |G(S) - G(S^i)| \leq c_i,$$

*then*

$$\forall \epsilon{>}0, \quad \mathbb{P}_S\left[\Big|G(S) - \mathbb{E}_S\big[G(S)\big]\Big| \geq \epsilon\right] \leq 2\exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2}\right)$$

*where $\mathbb{P}_S$ denotes the probability with respect to the random draw of the dataset $S$ from $\mathcal{D}^m$.*

Then, one can derive the following theorem.

**Theorem 3** ([Bellet et al., 2015] Th. 8.11). *Let $S$ be a dataset of $m$ randomly selected training examples and $\mathbf{M}$ be the PSD matrix learned from an algorithm with stability $\beta$. Assuming that the loss $\ell$ is $q$-Lipschitz and $(\sigma, p)$-admissible, with probability at least $1 - \delta$ over the random choice of $S \sim \mathcal{D}^m$, we have the following bound on the true risk $R(\mathbf{M})$*

$$R(\mathbf{M}) \leq \widehat{R}(\mathbf{M}) + 2\beta + \Big(2m\beta + 2(2\sigma + p)\Big)\sqrt{\frac{\ln 2/\delta}{2m}}.$$

This kind of generalization bound has two advantages: *(i)* unlike Vapnik-Chervonenkis dimension-based bounds [Vapnik and Chervonenkis, 1971], it takes into consideration properties of the algorithm, and *(ii)* it offers tools to deal with the fact that the pairs of examples are usually not drawn *i.i.d.* from $\mathcal{D} \times \mathcal{D}$ [Bellet et al., 2015]. In the rest of this section, we first show that our loss is $q$-Lipschitz; then, we prove that our algorithm is stable, and finally, we derive a generalization bound on its true risk using the McDiarmid inequality. In the following, we assume that the norm of any example is upper-bounded by a constant, *i.e.*, $\forall \boldsymbol{x} \in \mathbb{R}^d$, $\|\boldsymbol{x}\| \leq B$.

**Lemma 1.** *Let $\mathbf{M}$, $\mathbf{M}'$ be any matrices and $(\mathbf{z}, \mathbf{z}')$ any pair of labeled examples, then the loss $\ell$, as defined in Equation* (2.2)*, is $q$-Lipschitz w.r.t. its first argument, i.e., we have*

$$\Big|\ell(\mathbf{M}, \mathbf{z}, \mathbf{z}') - \ell(\mathbf{M}', \mathbf{z}, \mathbf{z}')\Big| \leq q\|\mathbf{M} - \mathbf{M}'\|_F,$$

*with $q = 4B^2$.*

*Proof.* Let $\mathbf{z} = (\boldsymbol{x}, y)$, $\mathbf{z}' = (\boldsymbol{x}', y')$ be two examples and $\mathbf{M}$, $\mathbf{M}'$ be two matrices. If $y = +1$ and $y' = +1$ we have

$$\begin{aligned}&\Big|\ell(\mathbf{M}, \mathbf{z}, \mathbf{z}') - \ell(\mathbf{M}', \mathbf{z}, \mathbf{z}')\Big|\\ =\ &\Big|a\ell_1(\mathbf{M}, \mathbf{z}, \mathbf{z}') - a\ell_1(\mathbf{M}', \mathbf{z}, \mathbf{z}')\Big|\end{aligned}$$

$$
\begin{aligned}
&= \Big|a[d^2_{\mathbf{M}}(\boldsymbol{x},\boldsymbol{x}')-1]_+ - a[d^2_{\mathbf{M}'}(\boldsymbol{x},\boldsymbol{x}')-1]_+\Big| \\
&= a\Big|[d^2_{\mathbf{M}}(\boldsymbol{x},\boldsymbol{x}')-1]_+ - [d^2_{\mathbf{M}'}(\boldsymbol{x},\boldsymbol{x}')-1]_+\Big| && (2.5)\\
&\leq \Big|[d^2_{\mathbf{M}}(\boldsymbol{x},\boldsymbol{x}')-1]_+ - [d^2_{\mathbf{M}'}(\boldsymbol{x},\boldsymbol{x}')-1]_+\Big| && (2.6)\\
&\leq \Big|d^2_{\mathbf{M}}(\boldsymbol{x},\boldsymbol{x}')-1 - d^2_{\mathbf{M}'}(\boldsymbol{x},\boldsymbol{x}')+1\Big| && (2.7)\\
&= \Big|d^2_{\mathbf{M}}(\boldsymbol{x},\boldsymbol{x}') - d^2_{\mathbf{M}'}(\boldsymbol{x},\boldsymbol{x}')\Big| \\
&= \big\langle \boldsymbol{x}-\boldsymbol{x}', (\mathbf{M}-\mathbf{M}')(\boldsymbol{x}-\boldsymbol{x}')\big\rangle \\
&\leq \|\boldsymbol{x}-\boldsymbol{x}'\|\|(\mathbf{M}-\mathbf{M}')(\boldsymbol{x}-\boldsymbol{x}')\| && (2.8)\\
&\leq \|\boldsymbol{x}-\boldsymbol{x}'\|\|\boldsymbol{x}-\boldsymbol{x}'\|\|\mathbf{M}-\mathbf{M}'\|_F && (2.9)\\
&\leq \Big(\|\boldsymbol{x}\|+\|-\boldsymbol{x}'\|\Big)\Big(\|\boldsymbol{x}\|+\|-\boldsymbol{x}'\|\Big)\|\mathbf{M}-\mathbf{M}'\|_F && (2.10)\\
&\leq 4B^2\|\mathbf{M}-\mathbf{M}'\|_F\,. && (2.11)
\end{aligned}
$$

Lines (2.5) and (2.6) come from the fact that $a \in [0,1]$, line (2.7) is obtained by using the fact that $\forall u \in \mathbb{R}, \forall v \in \mathbb{R}$ we have $|\max(0,u)-\max(0,v)| \leq |u-v|$, Line (2.8) from the Cauchy-Schwarz inequality, Lines (2.9) and (2.10) from norm properties and Line (2.11) from the fact that we assumed that $\forall \boldsymbol{x} \in \mathbb{R}^d, \|\boldsymbol{x}\| \leq B$. Similarly when $y=+1$ and $y'=-1$, or $y=-1$ and $y'=+1$ or $y=-1$ and $y'=-1$, we obtain a bound of $4B^2\|\mathbf{M}-\mathbf{M}'\|_F$. Thus for any pair of labeled examples $(\mathbf{z},\mathbf{z}')$, we have $\Big|\ell(\mathbf{M},\mathbf{z},\mathbf{z}')-\ell(\mathbf{M}',\mathbf{z},\mathbf{z}')\Big| \leq q\|\mathbf{M}-\mathbf{M}'\|_F$ with $q=4B^2$. □

Let $\mathbf{M}$ be the matrix learned from $S$ and $Sim^+$, $Dis^+$, $Dis^-$ and $Sim^-$ be the subsets of pairs coming from $S \times S$ as described in Section 2.3. The true and empirical losses are respectively defined as:

$$
\begin{aligned}
R(\mathbf{M}) =& \mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{D},\mathbf{z}'\sim\mathcal{D}} \ell(\mathbf{M},\mathbf{z},\mathbf{z}') \\
\text{and}\quad \widehat{R}(\mathbf{M}) =& \frac{1}{m^2}\Bigg(\sum_{(\mathbf{z},\mathbf{z}')\in Sim^+} a\ell_1(\mathbf{M},\mathbf{z},\mathbf{z}') + \sum_{(\mathbf{z},\mathbf{z}')\in Sim^-}(1-a)\ell_1(\mathbf{M},\mathbf{z},\mathbf{z}') + \\
&\sum_{(\mathbf{z},\mathbf{z}')\in Dis^+} b\ell_2(\mathbf{M},\mathbf{z},\mathbf{z}') + \sum_{(\mathbf{z},\mathbf{z}')\in Dis^-}(1-b)\ell_2(\mathbf{M},\mathbf{z},\mathbf{z}')\Bigg).
\end{aligned}
$$

where $\mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{D},\mathbf{z}'\sim\mathcal{D}}$ denotes the expectation with respect to the random draw of $\mathbf{z}$ and $\mathbf{z}'$ according to $\mathcal{D}$. Thus Equation (2.3) can be reformulated as:

$$\min_{\mathbf{M}\succeq 0} F(\mathbf{M}) = \widehat{R}(\mathbf{M}) + \lambda\|\mathbf{M}-\mathbf{I}\|_F^2.$$

**Uniform Stability of IML.** We now proceed to show that our algorithm satisfies Definition 4. For that purpose, we introduce a lemma similar to Lemma 20 from Bousquet and Elisseeff [2002] and Lemma 8.6 from Bellet et al. [2015].

**Lemma 2.** *Let matrices $\mathbf{M}^*$ and $\mathbf{M}^{*i}$ be the minimizers of $F$ on $S$ and $S^i$ respectively. Let $\Delta\mathbf{M}^* = \mathbf{M}^{*i} - \mathbf{M}^*$ and $\rho = \frac{m^+}{m}$ the proportion of minority examples. Then for any $t \in [0,1]$*

*we have*

$$\begin{aligned}&\|\mathbf{M}^*{-}\mathbf{I}\|_F^2 - \|\mathbf{M}^*{+}t\Delta\mathbf{M}^*{-}\mathbf{I}\|_F^2 + \|\mathbf{M}^{*i}{-}\mathbf{I}\|_F^2 - \|\mathbf{M}^{*i}{-}t\Delta\mathbf{M}^*{-}\mathbf{I}\|_F^2\\ \leq\;& \left(\frac{a(2\rho-1)+2(1-\rho)}{\lambda m}\right) 2qt\|\mathbf{M}^{*i}-\mathbf{M}^*\|_F\,.\end{aligned}$$

*Proof.* Firstly, a convex function $f$ satisfies for all $u, v$ and for all $t \in [0,1]$,

$$f(u+t(v-u)) - f(u) \leq tf(v) - tf(u).$$

Let $\widehat{R}^i(\mathbf{M}^*)$ be the empirical risk over $S^i$. Since $\widehat{R}^i(\mathbf{M}^*)$ is convex, we have

$$\widehat{R}^i(\mathbf{M}^* + t\Delta\mathbf{M}^*) - \widehat{R}^i(\mathbf{M}^*) \leq t\widehat{R}^i(\mathbf{M}^{*i}) - t\widehat{R}^i(\mathbf{M}^*). \tag{2.12}$$

By switching $\mathbf{M}^*$ and $\mathbf{M}^{*i}$ we have

$$\widehat{R}^i(\mathbf{M}^{*i} - t\Delta\mathbf{M}^*) - \widehat{R}^i(\mathbf{M}^{*i}) \leq t\widehat{R}^i(\mathbf{M}^*) - t\widehat{R}^i(\mathbf{M}^{*i}). \tag{2.13}$$

Summing Equation (2.12) and Equation (2.13) gives

$$\widehat{R}^i(\mathbf{M}^* + t\Delta\mathbf{M}^*) - \widehat{R}^i(\mathbf{M}^*) + \widehat{R}^i(\mathbf{M}^{*i} - t\Delta\mathbf{M}^*) - \widehat{R}^i(\mathbf{M}^{*i}) \leq 0. \tag{2.14}$$

We denote $F_S$ and $F_{S^i}$ the function $F$ to minimize respectively on $S$ and on $S^i$. Since $\mathbf{M}^*$ and $\mathbf{M}^{*i}$ are the minimizers of $F_S$ and $F_{S^i}$, we have

$$F_S(\mathbf{M}^*) - F_S(\mathbf{M}^* + t\Delta\mathbf{M}^*) \;\leq\; 0, \tag{2.15}$$

$$\text{and}\quad F_{S^i}(\mathbf{M}^{*i}) - F_{S^i}(\mathbf{M}^{*i} - t\Delta\mathbf{M}^*) \;\leq\; 0\,. \tag{2.16}$$

Summing Equations (2.15) and Equation (2.16) gives

$$F_S(\mathbf{M}^*) - F_S(\mathbf{M}^* + t\Delta\mathbf{M}^*) + F_{S^i}(\mathbf{M}^{*i}) - F_{S^i}(\mathbf{M}^{*i} - t\Delta\mathbf{M}^*) \;\leq\; 0\,. \tag{2.17}$$

Replacing $F$ in Equation (2.17) by its definition gives

$$\begin{aligned}&\widehat{R}(\mathbf{M}^*) - \widehat{R}(\mathbf{M}^*{+}t\Delta\mathbf{M}^*) + \widehat{R}^i(\mathbf{M}^{*i}) - \widehat{R}^i(\mathbf{M}^{*i}{-}t\Delta\mathbf{M}^*) + \lambda\Big(\|\mathbf{M}^*{-}\mathbf{I}\|_F^2\\ &- \|\mathbf{M}^*{+}t\Delta\mathbf{M}^*{-}\mathbf{I}\|_F^2 + \|\mathbf{M}^{*i}{-}\mathbf{I}\|_F^2 - \|\mathbf{M}^{*i}{-}t\Delta\mathbf{M}^*{-}\mathbf{I}\|_F^2\Big) \leq 0.\end{aligned} \tag{2.18}$$

Adding Equation (2.14) to Equation (2.18) gives

$$\begin{aligned}&\lambda\Big(\|\mathbf{M}^*{-}\mathbf{I}\|_F^2 - \|\mathbf{M}^*{+}t\Delta\mathbf{M}^*{-}\mathbf{I}\|_F^2 + \|\mathbf{M}^{*i}{-}\mathbf{I}\|_F^2 - \|\mathbf{M}^{*i}{-}t\Delta\mathbf{M}^*{-}\mathbf{I}\|_F^2\Big)\\ &\leq\; -\widehat{R}(\mathbf{M}^*) + \widehat{R}(\mathbf{M}^* + t\Delta\mathbf{M}^*) - \widehat{R}^i(\mathbf{M}^* + t\Delta\mathbf{M}^*) + \widehat{R}^i(\mathbf{M}^*).\end{aligned} \tag{2.19}$$

The left-hand side of Equation (2.19) corresponds to the left-hand side of the lemma multiplied by $\lambda$. As described in Section 2.3, we consider subsets of $S \times S$ which are *Sim*$^+$, *Dis*$^+$, *Dis*$^-$ and *Sim*$^-$. We do the same for $S^i \times S^i$ to obtain *Sim*$^{i+}$, *Dis*$^{i+}$, *Dis*$^{i-}$ and *Sim*$^{i-}$. Let $U, U^i$, $V, V^i$, $W, W^i$ and $X, X^i$ be the sets of different pairs respectively between *Sim*$^+$ and *Sim*$^{i+}$, between *Dis*$^+$ and *Dis*$^{i+}$, between *Dis*$^-$ and *Dis*$^{i-}$ and between *Sim*$^-$ and *Sim*$^{i-}$. Since the other pairs cancel each other, by bounding the right hand side we have

$$\widehat{R}(\mathbf{M}^* + t\Delta\mathbf{M}^*) - \widehat{R}(\mathbf{M}^*) - \widehat{R}^i(\mathbf{M}^* + t\Delta\mathbf{M}^*) + \widehat{R}^i(\mathbf{M}^*)$$

$$
\begin{aligned}
&\leq \left|\widehat{R}(\mathbf{M}^*+t\Delta\mathbf{M}^*)-\widehat{R}(\mathbf{M}^*)-\widehat{R}^i(\mathbf{M}^*+t\Delta\mathbf{M}^*)+\widehat{R}^i(\mathbf{M}^*)\right| \\
&= \left|\frac{a}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in Sim^+}\big(\ell_1(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_1(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\big)+\right. \\
&\quad \frac{b}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in Dis^+}\big(\ell_2(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_2(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\big)+ \\
&\quad \frac{1-b}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in Dis^-}\big(\ell_2(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_2(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\big)+ \\
&\quad \frac{1-a}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in Sim^-}\big(\ell_1(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_1(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\big)+ \\
&\quad \frac{a}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in Sim^{i+}}\big(-\ell_1(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')+\ell_1(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\big)+ \\
&\quad \frac{b}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in Dis^{i+}}\big(-\ell_2(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')+\ell_2(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\big)+ \\
&\quad \frac{1-b}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in Dis^{i-}}\big(-\ell_2(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')+\ell_2(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\big)+ \\
&\quad \left.\frac{1-a}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in Sim^{i-}}\big(-\ell_1(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')+\ell_1(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\big)\right| \\
&\leq \frac{a}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in U}\left|\ell_1(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_1(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\right|+ \\
&\quad \frac{b}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in V}\left|\ell_2(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_2(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\right|+ \\
&\quad \frac{1-b}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in W}\left|\ell_2(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_2(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\right|+ \\
&\quad \frac{1-a}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in X}\left|\ell_1(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_1(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\right|+ \\
&\quad \frac{a}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in U^i}\left|\ell_1(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_1(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\right|+ \\
&\quad \frac{b}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in V^i}\left|\ell_2(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_2(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\right|+ \\
&\quad \frac{1-b}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in W^i}\left|\ell_2(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_2(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\right|+ \\
&\quad \frac{1-a}{m^2}\sum_{(\mathbf{z},\mathbf{z}')\in X^i}\left|\ell_1(\mathbf{M}^*+t\Delta\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\ell_1(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\right|.
\end{aligned}
$$

Depending on the label of the example replaced and the label of the substitute instance, the sets $U$, $U^i$, $V$, $V^i$, $W$, $W^i$, $X$ and $X^i$ will contain a different number of pairs. This number is at most $2m^+$ for $|U|+|U^i|$ (by taking in $Sim^+$ all the $m^+m^+$ possible pairs), at most $2m^-$ for

$|V|+|V^i|$ and $|W|+|W^i|$ (by taking in $Dis^+$ and $Dis^-$ all the $m^+m^-$ possible pairs) and at most $2m^-$ for $|X|+|X^i|$ (by taking in $Sim^-$ all the $m^-m^-$ possible pairs). Using the $q$-Lipschitz property from Lemma 1 we obtain

$$\begin{aligned}
&|\widehat{R}(\mathbf{M}^*+t\Delta\mathbf{M}^*)-\widehat{R}(\mathbf{M}^*)-\widehat{R}^i(\mathbf{M}^*+t\Delta\mathbf{M}^*)+\widehat{R}^i(\mathbf{M}^*)|\\
\leq\ & \left(\frac{am^+ + (b+1-b+1-a)m^-}{m^2}\right) 2qt\|\mathbf{M}^{*i}-\mathbf{M}^*\|_F\\
=\ & \left(\frac{\rho a + (2-a)(1-\rho)}{m}\right) 2qt\|\mathbf{M}^{*i}-\mathbf{M}^*\|_F\\
=\ & \left(\frac{a(2\rho-1) + 2(1-\rho)}{m}\right) 2qt\|\mathbf{M}^{*i}-\mathbf{M}^*\|_F\,.
\end{aligned}$$

Combining with Equation (2.19) gives

$$\begin{aligned}
&\|\mathbf{M}^*-\mathbf{I}\|_F^2 - \|\mathbf{M}^*+t\Delta\mathbf{M}^*-\mathbf{I}\|_F^2 + \|\mathbf{M}^{*i}-\mathbf{I}\|_F^2 - \|\mathbf{M}^{*i}-t\Delta\mathbf{M}^*-\mathbf{I}\|_F^2\\
\leq\ & \left(\frac{a(2\rho-1) + 2(1-\rho)}{\lambda m}\right) 2qt\|\mathbf{M}^{*i}-\mathbf{M}^*\|_F\,.
\end{aligned}$$

□

The parameter $b$ of Equation (2.3) does not appear in this Lemma. While in the experiments, in order to scale to large datasets, the pairs will be generated by using the k-neighborhood of the examples, we derived the proof in Lemma 2 by using all the pairs, that allowed us to get rid of the parameter $b$. This enables us to provide a more general result that does not depend on additional parameters (here the $k$ of the $k$-nearest-neighbor rule).

Along with the $q$-Lipschitz property of Lemma 1, Lemma 2 allows us to prove that **IML** is stable.

**Lemma 3. IML** *has uniform stability with*

$$\beta = \frac{2q^2\Big(a(2\rho-1)+2(1-\rho)\Big)}{\lambda m}.$$

*Proof.* By setting $t=\frac{1}{2}$ in Lemma 2 we have

$$\begin{aligned}
&\|\mathbf{M}^*-\mathbf{I}\|_F^2 - \|\tfrac{1}{2}(\mathbf{M}^{*i}+\mathbf{M}^*)-\mathbf{I}\|_F^2 + \|\mathbf{M}^{*i}-\mathbf{I}\|_F^2 - \left\|\tfrac{1}{2}(\mathbf{M}^{*i}+\mathbf{M}^*)-\mathbf{I}\right\|_F^2\\
&= \|\mathbf{M}^*-\mathbf{I}\|_F^2 + \|\mathbf{M}^{*i}-\mathbf{I}\|_F^2 - 2\left\|\tfrac{1}{2}(\mathbf{M}^{*i}+\mathbf{M}^*)-\mathbf{I}\right\|_F^2\\
&= \|\mathbf{M}^*-\mathbf{I}\|_F^2 + \|\mathbf{M}^{*i}-\mathbf{I}\|_F^2 - 2\left\|\tfrac{1}{2}(\mathbf{M}^{*i}+\mathbf{M}^*-2\mathbf{I})\right\|_F^2\\
&= \|\mathbf{M}^*-\mathbf{I}\|_F^2 + \|\mathbf{M}^{*i}-\mathbf{I}\|_F^2 - 2\tfrac{1}{4}\left\|\mathbf{M}^{*i}+\mathbf{M}^*-2\mathbf{I}\right\|_F^2\\
&= \|\mathbf{M}^*-\mathbf{I}\|_F^2 + \|\mathbf{M}^{*i}-\mathbf{I}\|_F^2 - \tfrac{1}{2}\left\|\mathbf{M}^*-\mathbf{I}+\mathbf{M}^{*i}-\mathbf{I}\right\|_F^2\\
&= \langle\mathbf{M}^*-\mathbf{I},\mathbf{M}^*-\mathbf{I}\rangle + \left\langle\mathbf{M}^{*i}-\mathbf{I},\mathbf{M}^{*i}-\mathbf{I}\right\rangle -\\
&\quad \tfrac{1}{2}\left\langle\mathbf{M}^*-\mathbf{I}+\mathbf{M}^{*i}-\mathbf{I},\mathbf{M}^*-\mathbf{I}+\mathbf{M}^{*i}-\mathbf{I}\right\rangle \qquad (2.20)\\
&= \tfrac{1}{2}\Big(\langle\mathbf{M}^*,\mathbf{M}^*\rangle + \left\langle\mathbf{M}^{*i},\mathbf{M}^{*i}\right\rangle - \left\langle\mathbf{M}^*,\mathbf{M}^{*i}\right\rangle - \left\langle\mathbf{M}^{*i},\mathbf{M}^*\right\rangle\Big) \qquad (2.21)\\
&= \tfrac{1}{2}\Big(\langle-\mathbf{M}^*,-\mathbf{M}^*\rangle + \left\langle\mathbf{M}^{*i},\mathbf{M}^{*i}\right\rangle + \left\langle-\mathbf{M}^*,\mathbf{M}^{*i}\right\rangle + \left\langle\mathbf{M}^{*i},-\mathbf{M}^*\right\rangle\Big)
\end{aligned}$$

$$
\begin{aligned}
&= \tfrac{1}{2}\left\langle \mathbf{M}^{*i} - \mathbf{M}^*, \mathbf{M}^{*i} - \mathbf{M}^* \right\rangle \\
&= \tfrac{1}{2}\|\mathbf{M}^{*i} - \mathbf{M}^*\|_F^2 \,,
\end{aligned}
$$

where from Equation (2.20) to Equation (2.21) we develop using $\langle A+B, C+D\rangle = \langle A, C\rangle + \langle A, D\rangle + \langle B, C\rangle + \langle B, D\rangle$, with $\langle \cdot, \cdot\rangle$ is the Frobenius inner product.

To resume, from Lemma 2 with $t = \frac{1}{2}$ we have

$$\|\mathbf{M}^{*i} - \mathbf{M}^*\|_F \leq \left(\frac{a(2\rho-1) + 2(1-\rho)}{\lambda m}\right) 2q\,.$$

Using the $q$-Lipschitz property of $\ell$ we obtain that

$$
\begin{aligned}
&|\ell(\mathbf{M}^*, \mathbf{z}, \mathbf{z}') - \ell(\mathbf{M}^{*i}, \mathbf{z}, \mathbf{z}')| \\
&\leq q\left(\frac{a(2\rho-1) + 2(1-\rho)}{\lambda m}\right) 2q \\
&= \frac{2q^2\Big(a(2\rho-1) + 2(1-\rho)\Big)}{\lambda m} \\
&= \beta\,.
\end{aligned}
$$

□

**Derivation of the main result.** To prove our bound, we follow the derivation of Theorem 8.11 from Bellet et al. [2015]. We first provide two lemmas, and then we use them in conjunction with the McDiarmid inequality to derive a generalization bound.

First, we introduce a lemma that bounds the difference on the empirical risk over $S$ and $S^i$.

**Lemma 4.** *Let* $\mathbf{M}^*$ *be the optimal solution of Equation* (2.3)*. We have*

$$
\begin{aligned}
&\left|\widehat{R}(\mathbf{M}^*) - \widehat{R^i}(\mathbf{M}^*)\right| \\
\leq &\frac{2\Big(a(2\rho-1) + 1 - \rho\Big)4B^2\|\mathbf{M}^*\|_F + 2(1-\rho)(1+\kappa)}{m}.
\end{aligned}
$$

*where* $\widehat{R}$ *and* $\widehat{R}^i$ *denote respectively the empirical risk over* $S$ *and* $S^i$*.*

*Proof.* Let $\mathbf{M}$ be the optimal solution of Equation (2.3) and $S$ a training set. In a similar way as in the proof of Lemma 2, let $U, U^i$, $V, V^i$, $W, W^i$ and $X, X^i$ be the sets of different pairs respectively between $Sim^+$ and $Sim^{i+}$, between $Dis^+$ and $Dis^{i+}$, between $Dis^-$ and $Dis^{i-}$ and between $Sim^-$ and $Sim^{i-}$. Then

$$
\begin{aligned}
&\left|\widehat{R}(\mathbf{M}^*) - \widehat{R^i}(\mathbf{M}^*)\right| \\
= &\left|\frac{a}{m^2}\left(\sum_{(\mathbf{z},\mathbf{z}')\in U} \ell_1(\mathbf{M}^*, \mathbf{z}, \mathbf{z}') - \sum_{(\mathbf{z},\mathbf{z}')\in U^i} \ell_1(\mathbf{M}^*, \mathbf{z}, \mathbf{z}')\right) + \right. \\
&\frac{b}{m^2}\left(\sum_{(\mathbf{z},\mathbf{z}')\in V} \ell_2(\mathbf{M}^*, \mathbf{z}, \mathbf{z}') - \sum_{(\mathbf{z},\mathbf{z}')\in V^i} \ell_2(\mathbf{M}^*, \mathbf{z}, \mathbf{z}')\right) +
\end{aligned}
$$

$$
\begin{aligned}
&\frac{1-b}{m^2}\left(\sum_{(\mathbf{z},\mathbf{z}')\in W}\ell_2(\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\sum_{(\mathbf{z},\mathbf{z}')\in W^i}\ell_2(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\right)+\\
&\frac{1-a}{m^2}\left(\sum_{(\mathbf{z},\mathbf{z}')\in X}\ell_1(\mathbf{M}^*,\mathbf{z},\mathbf{z}')-\sum_{(\mathbf{z},\mathbf{z}')\in X^i}\ell_1(\mathbf{M}^*,\mathbf{z},\mathbf{z}')\right)\Bigg|\\
\le\ &\frac{a}{m^2}\left(\left|\sum_{(\mathbf{z},\mathbf{z}')\in U}4B^2\|\mathbf{M}^*\|_F\right|+\left|\sum_{(\mathbf{z},\mathbf{z}')\in U^i}4B^2\|\mathbf{M}^*\|_F\right|\right)+\\
&\frac{b}{m^2}\left(\left|\sum_{(\mathbf{z},\mathbf{z}')\in V}(1+\kappa)\right|+\left|\sum_{(\mathbf{z},\mathbf{z}')\in V^i}(1+\kappa)\right|\right)+\\
&\frac{1-b}{m^2}\left(\left|\sum_{(\mathbf{z},\mathbf{z}')\in W}(1+\kappa)\right|+\left|\sum_{(\mathbf{z},\mathbf{z}')\in W^i}(1+\kappa)\right|\right)+\\
&\frac{1-a}{m^2}\left(\left|\sum_{(\mathbf{z},\mathbf{z}')\in X}4B^2\|\mathbf{M}^*\|_F\right|+\left|\sum_{(\mathbf{z},\mathbf{z}')\in X^i}4B^2\|\mathbf{M}^*\|_F\right|\right)
\end{aligned}
\tag{2.22}
$$

where Equation (2.22) comes from the fact that $\sup_{\mathbf{z}_1,\mathbf{z}_2}\ell_1(\mathbf{M}^*,\mathbf{z}_1,\mathbf{z}_2)\le 4B^2\|\mathbf{M}^*\|_F$ (obtained similarly as in the Proof of the $q$-Lipschitz property) and that $\sup_{\mathbf{z}_1,\mathbf{z}_2}\ell_2(\mathbf{M}^*,\mathbf{z}_1,\mathbf{z}_2)\le 1+\kappa$ (because the Mahalanobis distance is always positive).

As $4B^2\|\mathbf{M}^*\|_F$ and $(1+\kappa)$ are always positive, we have

$$
\begin{aligned}
&\left|\widehat{R}(\mathbf{M}^*)-\widehat{R^i}(\mathbf{M}^*)\right|\\
\le\ &\frac{\Big(a\big(|U|+|U^i|\big)+(1-a)\big(|X|+|X^i|\big)\Big)4B^2\|\mathbf{M}^*\|_F}{m^2}+\\
&\frac{\Big(b\big(|V|+|V^i|\big)+(1-b)\big(|W|+|W^i|\big)\Big)(1+\kappa)}{m^2}.
\end{aligned}
$$

As in the proof of Lemma 2, the number of pairs in the subsets is at most $m^+$ for $U$ and $U^i$, and at most $m^-$ for $V$, $V^i$, $W$, $W^i$, $X$ and $X^i$. Thus for any labeled example replaced by any labeled example, we have

$$
\begin{aligned}
&\left|\widehat{R}(\mathbf{M}^*)-\widehat{R^i}(\mathbf{M}^*)\right|\\
\le\ &\frac{2(am^++(1-a)m^-)4B^2\|\mathbf{M}^*\|_F+2m^-(b+1-b)(1+\kappa)}{m^2}\\
=\ &\frac{2\Big(a(2\rho-1)+1-\rho\Big)4B^2\|\mathbf{M}^*\|_F+2(1-\rho)(1+\kappa)}{m}.
\end{aligned}
$$

□

We recall a useful Lemma coming from Bellet et al. [2015] and Bousquet and Elisseeff [2002]:

**Lemma 5** ([Bellet et al., 2015] Lemma 8.9, [Bousquet and Elisseeff, 2002] in Proof of Theorem 12)**.** *For any learning method of estimation error $R(\mathbf{M})-\widehat{R}(\mathbf{M})$ and satisfying uniform stability in $\beta$ we have*

$$\mathbb{E}_S[R(\mathbf{M})-\widehat{R}(\mathbf{M})]\le 2\beta.$$

Using Lemma 4 and the stability of Lemma 3, we introduce a lemma similar to Lemma 8.10 by Bellet et al. [2015].

**Lemma 6.** *Let matrices* $\mathbf{M}^*$ *and* $\mathbf{M}^{*i}$ *be the minimizers of* $F$ *on* $S$ *and* $S^i$ *respectively. As* **IML** *has stability* $\beta$*, we have*

$$\left|R(\mathbf{M}^*) - \widehat{R}(\mathbf{M}^*) - \Big(R(\mathbf{M}^{*i}) - \widehat{R}^i(\mathbf{M}^{*i})\Big)\right| \leq \frac{2m\beta + D}{m} = c_i,$$

*with*
$D = 2\Big(a(2\rho - 1) + 1 - \rho\Big)4B^2\|\mathbf{M}^*\|_F + 2(1-\rho)(1{+}\kappa)$.

*Proof.*

$$\begin{aligned}
&\left|R(\mathbf{M}^*) - \widehat{R}(\mathbf{M}^*) - \Big(R(\mathbf{M}^{*i}) - \widehat{R}^i(\mathbf{M}^{*i})\Big)\right| \\
&= \left|R(\mathbf{M}^*) - \widehat{R}(\mathbf{M}^*) - R(\mathbf{M}^{*i}) + \widehat{R}^i(\mathbf{M}^{*i}) + \widehat{R}(\mathbf{M}^{*i}) - \widehat{R}(\mathbf{M}^{*i})\right| \\
&= \left|R(\mathbf{M}^*) - R(\mathbf{M}^{*i}) + \widehat{R}(\mathbf{M}^{*i}) - \widehat{R}(\mathbf{M}^*) + \widehat{R}^i(\mathbf{M}^{*i}) - \widehat{R}(\mathbf{M}^{*i})\right| \\
&\leq \left|R(\mathbf{M}^*) - R(\mathbf{M}^{*i})\right| + \left|\widehat{R}(\mathbf{M}^{*i}) - \widehat{R}(\mathbf{M}^*)\right| + \left|\widehat{R}^i(\mathbf{M}^{*i}) - \widehat{R}(\mathbf{M}^{*i})\right| \\
&\leq \beta + \beta + \frac{D}{m} \\
&= \frac{2m\beta + D}{m}
\end{aligned}$$

with
$D = 2\Big(a(2\rho - 1) + 1 - \rho\Big)4B^2\|\mathbf{M}^*\|_F + 2(1-\rho)(1{+}\kappa)$. □

We are now able to state our main result.

**Theorem 4** (Generalization bound for **IML**). *Let* $S$ *be a dataset of* $m = m^+ + m^-$ *randomly selected training examples with* $\rho = \frac{m^+}{m}$ *the proportion of minority examples and let* $\mathbf{M}^*$ *be the optimal solution learned from Equation* (2.3) *having stability* $\beta$*. With probability at least* $1 - \delta$ *over the random choice of* $S \sim \mathcal{D}^m$*, we have*

$$\begin{aligned}
R(\mathbf{M}^*) &\leq \widehat{R}(\mathbf{M}^*) + 2\beta + (2m\beta + D)\sqrt{\frac{\ln 2/\delta}{2m}} \\
\textit{with } \beta &= \frac{2q^2\Big(a(2\rho-1) + 2(1-\rho)\Big)}{\lambda m} \\
\textit{and } D &= 2\Big(a(2\rho - 1) + 1 - \rho\Big)4B^2\|\mathbf{M}^*\|_F + 2(1-\rho)(1{+}\kappa)\,.
\end{aligned}$$

*Proof.* From Lemma 6, we know that the variable $G(S) = R(\mathbf{M}^*) - \widehat{R}(\mathbf{M}^*)$ satisfies the condition of the McDiarmid inequality with the same $c_i = \frac{2m\beta+D}{m}$ $\forall i \in \{1, \ldots, m\}$. By applying the inequality we have

$$\mathbb{P}_S\Big[\big|G(S) - \mathbb{E}_S[G(S)]\big| \geq \epsilon\Big] \;\leq\; 2\exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2}\right).$$

By setting $\delta$ to the right hand side of the previous inequality we obtain

$$\begin{aligned}
&\delta = 2\exp\left(\frac{-2\epsilon^2}{mc_i^2}\right)\\
\iff &\ln\delta/2 = \frac{-2\epsilon^2 m^2}{m(2m\beta+D)^2}\\
\iff &\frac{\ln 2/\delta}{2m} = \frac{\epsilon^2}{(2m\beta+D)^2}\\
\iff &\epsilon = (2m\beta+D)\sqrt{\frac{\ln 2/\delta}{2m}}.
\end{aligned}$$

Thus with probability at least $1-\delta$ we have

$$\begin{aligned}
&|G(S) - \mathbb{E}_S[G(S)]| \leq \epsilon\\
\iff &|G(S) - \mathbb{E}_S[G(S)]| \leq (2m\beta+D)\sqrt{\frac{\ln 2/\delta}{2m}}\\
\iff &G(S) \leq \mathbb{E}_S[G(S)] + (2m\beta+D)\sqrt{\frac{\ln 2/\delta}{2m}}\\
\iff &R(\mathbf{M}^*) \leq \widehat{R}(\mathbf{M}^*) + 2\beta + (2m\beta+D)\sqrt{\frac{\ln 2/\delta}{2m}}. \qquad (2.23)
\end{aligned}$$

where Equation (2.23) comes from Lemma 5. □

**Discussion.** The difference between our Theorem 4 and classic bounds of the form of Theorem 3 is that proportion of minority examples $\rho = \frac{m^+}{m}$ and the weight of the similar minority pairs $a$ appear in the two terms $\beta$ and $D$. Classic metric learning bounds are derived in a balanced setting where $\rho = 0.5$ (*i.e.*, positives and negatives are balanced) and where the parameters $a$ and $b$ are equal to 0.5. It is worth noting that plugging these values in our bound allows us to retrieve the constant $\beta = \frac{2q^2}{\lambda m}$ as derived in Theorem 8.7 by Bellet et al. [2015]. This means that our $\beta$ formulation is a generalization of the standard stability constants in metric learning. Regarding the term $D$, the decomposition into four terms allows us to get a tighter bound by a factor 4 with $D = 4B^2\|\mathbf{M}\|_F + (1+\kappa)$ while $D = 16B^2\|\mathbf{M}\|_F + 4(1+\kappa)$ by Bellet et al. [2015].

Another interesting interpretation of our bound is that when $\rho$ tends to 0, *i.e.*, the dataset is more and more imbalanced, a classic metric learning method (with $a = b = 0.5$) will converge slower. Indeed, in such a situation, we get a stability constant $\beta$ which would tend to $\frac{3q^2}{\lambda m} > \frac{2q^2}{\lambda m}$ while by parameterizing by $a$, we have $\frac{2q^2(-a+2)}{\lambda m}$. In this case, a value of $a$ close to 1 allows us to reduce the negative effect of the imbalance.

*Table 2.1: Description of the datasets (m: number of examples, d: number of features, c: number of classes) and the class chosen as positive (Label), its cardinality ($m^+$) and its percentage (%).*

| Name | m | d | c | Label | $m^+$ | % | Name | m | d | c | Label | $m^+$ | % |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| splice | 3175 | 60 | 2 | -1 | 1527 | 48.10% | glass | 214 | 11 | 6 | 1 | 70 | 32.71% |
| sonar | 208 | 60 | 2 | R | 97 | 46.64% | newthyroid | 215 | 5 | 3 | 2, 3 | 65 | 30.23% |
| balance | 625 | 4 | 3 | L | 288 | 46.08% | german | 1000 | 23 | 2 | 2 | 300 | 30.00% |
| australian | 690 | 14 | 2 | 1 | 307 | 44.49% | vehicle | 846 | 18 | 4 | van | 199 | 23.52% |
| heart | 270 | 13 | 2 | 2 | 120 | 44.44% | spectfheart | 267 | 44 | 2 | 0 | 55 | 20.60% |
| bupa | 345 | 6 | 2 | 1 | 145 | 42.03% | hayes | 160 | 4 | 3 | 3 | 31 | 19.38% |
| spambase | 4597 | 57 | 2 | 1 | 1812 | 39.42% | segmentation | 2310 | 19 | 7 | window | 330 | 14.29% |
| wdbc | 569 | 30 | 2 | M | 212 | 37.26% | abalone | 4177 | 10 | 28 | 8 | 568 | 13.60% |
| iono | 351 | 34 | 2 | b | 126 | 35.90% | yeast | 1484 | 8 | 10 | ME3 | 163 | 10.98% |
| pima | 768 | 8 | 2 | 1 | 268 | 34.90% | libras | 360 | 90 | 15 | 1 | 24 | 6.66% |
| wine | 178 | 13 | 3 | 1 | 59 | 33.15% | pageblocks | 5473 | 10 | 5 | 3, 4, 5 | 231 | 4.22% |

## 2.5 Experiments

### 2.5.1 Datasets

We provide here an empirical study of **IML** on 22 datasets coming mainly from the UCI[3] and Keel[4] repositories except for the 'SPLICE' dataset which comes from LIBSVM[5]. All datasets are normalized such that each feature has a mean of 0 and a variance of 1.

For the sake of simplicity, we have chosen binary datasets, described in Table 2.1 where the minority class is given by the columns "Label". **IML** can easily be generalized to multi-class problems by learning one metric per class in a standard "one-versus-all" strategy, and then applying a majority vote [Schölkopf et al., 1995, Vapnik, 1995].

### 2.5.2 Optimization details

Like most Mahalanobis metric learning algorithms, **IML** requires the learned matrix $\mathbf{M}$ to be PSD. There exist different methods to enforce the PSD constraint [Kulis, 2013]. A classic solution consists in performing a Projected Gradient Descent where one alternates a gradient descent step and a (costly) projection onto the cone of PSD matrices. The advantage is that the problem remains convex [Xing et al., 2003] *w.r.t.* $\mathbf{M}$, ensuring that one will attain the optimal solution of the problem by correctly setting the projection step in the gradient descent. Another solution [Weinberger and Saul, 2008] is based on the fact that if $\mathbf{M}$ is PSD, it can be rewritten as $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$. Therefore, instead of learning $\mathbf{M}$, one can enforce $\mathbf{M}$ to be PSD in a cheaper way by directly learning the projection matrix $\mathbf{L} \in \mathbb{R}^{r\times d}$ (where $r$ is the rank of $\mathbf{M}$). This can be done thanks to a gradient descent by computing the gradient of the problem *w.r.t.* $\mathbf{L}$ (instead

[3] `https://archive.ics.uci.edu/ml/datasets.html`

[4] `http://sci2s.ugr.es/keel/datasets.php`

[5] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#splice`

of **M**). The implementation[6] we propose is based on this latter approach [Weinberger and Saul, 2008] where we make use of the L-BFGS-B algorithm [Zhu et al., 1997] from the SciPy Python library to optimize our problem: it takes as input our initial point (the identity matrix), the optimization problem of Equation (2.3), and its gradient, then it performs a gradient descent that returns the projection matrix **L** minimizing Equation (2.3). To prevent us from tuning $r$ and finding the best $r$-dimensional projection space, we set $r = d$ in the experiments. Indeed, our main objective here is to learn a robust metric and not to perform dimensionality reduction.

As discussed at the end of Section 2.3, the pairs of examples considered by **IML** in its four terms are chosen using the nearest neighbors rule. Indeed, we noted experimentally that the algorithms using this strategy (**LMNN** [Weinberger and Saul, 2009], **IMLS** [Wang et al., 2018] and **IML**) perform better than the ones using a random selection strategy (**ITML** [Davis et al., 2007] and **GMML** [Zadeh et al., 2016]).

### 2.5.3 Experimental setup

All along our experiments, we use a 3NN classifier (like in **LMNN** [Weinberger and Saul, 2009]) after projection of the training and test data using the metric learned. The metrics considered in the comparative study are the Euclidean distance and the ones learned by **LMNN** [Weinberger and Saul, 2009], **ITML** [Davis et al., 2007], **GMML** [Zadeh et al., 2016], **IMLS** [Wang et al., 2018] and **IML**. For each dataset, we generate randomly 20 stratified splits of 70% training examples and 30% test data (same class proportions in training and test) and report the mean results over the 20 splits. The parameters are tuned to maximize the F1-measure by doing a 5-fold cross-validation on the training set through a grid search using the following parameter ranges: for **LMNN** and **IMLS**, $\mu \in \{0, 0.05, \dots, 1\}$ ($k$ is fixed to 3); for **ITML**, $\gamma \in \{2^{-10:10}\}$; for **GMML** $t \in \{0, 0.05, \dots, 1\}$; and for **IML** we fix $a = b = \frac{m^-}{m}$ and we tune $\kappa \in \{1, 10, 100, 1000, 10000\}$ and $\lambda \in \{0, 0.01, 0.1, 1, 10\}$ ($k$ is also set to 3).

### 2.5.4 Analysis of the results

**First experiment—without data pre-processing** We start by applying the experimental setup described above and we report the results in Table 2.2. On average, the F1-measure of 72.3% obtained by **IML** is the best in comparison to 70.8% for **LMNN** and **IMLS**, 70.1% for **ITML**, 69.3% for **GMML** and 67.3% for the Euclidean distance. Overall, **IML** shows also the best average rank of 1.52. **IML** generally gives better performances on the datasets considered no matter how much they are balanced or not. This means that our re-weighting scheme of the pairs can not only improve the performances in an imbalanced setting but can be also competitive in more classic scenarios.

**Second experiment—with data pre-processing** As previously said, to address imbalanced data issues, classic machine learning algorithms typically resort to over/under-sampling

[6] The code is available here: `https://leogautheron.github.io`

*Table 2.2: Average F1-measure ± standard deviation over 20 splits using different metric learning algorithms.*

| Dataset | Euclidean | LMNN | ITML | GMML | IMLS | IML |
|---|---|---|---|---|---|---|
| hayes | 44.9 ± 13.2 | **57.2** ± 12.5 | 55.4 ± 8.7 | 52.7 ± 10.8 | 57.2 ± 12.5 | 54.9 ± 9.2 |
| wine | 94.9 ± 2.2 | 96.0 ± 2.9 | 96.3 ± 3.3 | 95.3 ± 3.1 | 96.0 ± 2.9 | **96.6** ± 2.1 |
| sonar | 69.2 ± 5.3 | 70.6 ± 6.5 | 70.6 ± 5.9 | 69.1 ± 5.0 | 71.1 ± 6.7 | **74.6** ± 3.7 |
| glass | 66.0 ± 3.4 | 63.6 ± 5.2 | 62.6 ± 5.2 | **67.2** ± 3.6 | 63.6 ± 5.2 | 66.6 ± 4.3 |
| newthyroid | 83.4 ± 4.2 | 88.1 ± 5.2 | 89.8 ± 5.2 | 91.1 ± 2.5 | 88.1 ± 5.2 | **91.3** ± 2.6 |
| spectfheart | 34.8 ± 12.3 | 39.1 ± 8.4 | 34.4 ± 7.9 | 29.1 ± 11.4 | 38.6 ± 8.7 | **42.4** ± 8.7 |
| heart | 76.8 ± 2.1 | 74.8 ± 3.2 | 76.8 ± 2.9 | 76.9 ± 3.6 | 74.6 ± 3.1 | **77.1** ± 3.1 |
| bupa | 49.8 ± 4.4 | 50.1 ± 5.0 | 51.3 ± 4.8 | 52.0 ± 5.3 | 50.1 ± 5.0 | **52.5** ± 5.1 |
| iono | 67.8 ± 6.7 | 70.8 ± 3.9 | 73.4 ± 5.4 | 72.0 ± 5.4 | 71.0 ± 4.0 | **76.1** ± 2.9 |
| libras | 48.4 ± 15.1 | **68.3** ± 12.2 | 65.5 ± 15.3 | 56.1 ± 16.3 | 66.6 ± 10.3 | 67.9 ± 12.1 |
| wdbc | 94.2 ± 1.3 | 93.5 ± 1.7 | 94.3 ± 1.1 | 94.4 ± 1.3 | 93.4 ± 2.2 | **95.2** ± 1.1 |
| balance | 87.4 ± 1.8 | 89.8 ± 1.3 | **93.0** ± 1.4 | 90.3 ± 1.3 | 89.8 ± 1.3 | 90.6 ± 1.2 |
| australian | 79.9 ± 1.7 | 81.7 ± 2.0 | **82.0** ± 1.9 | 81.0 ± 2.6 | 81.4 ± 2.0 | 81.9 ± 1.8 |
| pima | 56.2 ± 1.9 | 55.9 ± 3.3 | **57.5** ± 3.0 | 56.7 ± 3.0 | 55.9 ± 3.3 | 57.2 ± 2.7 |
| vehicle | 80.5 ± 2.4 | **92.6** ± 1.0 | 90.2 ± 2.4 | 90.1 ± 1.7 | 92.5 ± 1.2 | 91.8 ± 1.9 |
| german | 35.3 ± 2.8 | 37.3 ± 3.9 | 37.4 ± 3.3 | 37.1 ± 3.3 | 37.8 ± 3.7 | **38.4** ± 3.5 |
| yeast | 73.2 ± 2.3 | 74.9 ± 2.8 | 74.2 ± 3.1 | 73.5 ± 2.6 | 74.5 ± 2.7 | **75.4** ± 2.4 |
| segmentation | 81.8 ± 2.4 | 85.3 ± 2.1 | 79.6 ± 3.0 | 80.8 ± 3.1 | 85.6 ± 2.3 | **86.0** ± 2.5 |
| splice | 76.3 ± 0.7 | 86.5 ± 0.8 | 79.7 ± 1.4 | 76.3 ± 1.3 | **88.0** ± 0.9 | 87.4 ± 0.6 |
| abalone | 22.6 ± 2.1 | 22.1 ± 2.1 | 21.2 ± 3.0 | 21.6 ± 1.7 | 22.1 ± 2.1 | **23.0** ± 1.9 |
| spambase | 85.3 ± 0.9 | 88.4 ± 0.8 | 87.8 ± 1.0 | 86.8 ± 0.8 | 88.7 ± 0.5 | **89.3** ± 0.8 |
| pageblocks | 71.9 ± 3.0 | 71.8 ± 3.2 | 69.7 ± 5.1 | **73.7** ± 2.9 | 71.8 ± 3.1 | 73.4 ± 2.6 |
| Mean | 67.3 ± 4.2 | 70.8 ± 4.1 | 70.1 ± 4.3 | 69.3 ± 4.2 | 70.8 ± 4.0 | **72.3** ± 3.5 |
| Average Rank | 5.00 | 3.57 | 3.57 | 4.00 | 3.35 | **1.52** |

techniques [Aggarwal, 2013] or create synthetic examples in the neighborhood of the minority class—*e.g.*, using SMOTE-like strategies [Chawla et al., 2002]. We now aim at studying the behavior of those methods when used as a pre-process of the metric learning procedures. We consider the results of Table 2.2 as baselines. We compare them to the performances obtained after performing prior to metric learning an over-sampling using SMOTE and a Random Under Sampling (RUS) strategy of the negative data. We use the implementations of these methods from the Python library *imbalanced-learn* [Lemaître et al., 2017].

The results obtained are reported in Table 2.3 for SMOTE and Table 2.4 RUS and were computed using the same training/test splits as in Table 2.2 and the same validation folds. Thus, they are comparable. In each of the three settings considered, **IML** obtains the best results showing that it is more appropriate for improving the F1-measure. SMOTE allows one to increase significantly the performances of all methods, while there is no gain with RUS in comparison with an approach without sampling.

This increase of performance suggests that SMOTE and **IML** are more complementary than competitors with different objectives. This intuition is supported by the following explanation. Learning a Mahalanobis distance boils down to optimizing an ellipsoid centered at each point

*Table 2.3: Same experiment as in Table 2.2 after having applied the SMOTE algorithm [Chawla et al., 2002] until $m^+$ reaches $m^-$.*

| Dataset | Euclidean | LMNN | ITML | GMML | IMLS | IML |
|---|---|---|---|---|---|---|
| hayes | 68.0 ± 6.8 | 64.4 ± 7.6 | 67.8 ± 7.8 | **69.5** ± 7.2 | 64.2 ± 7.6 | 68.6 ± 7.2 |
| wine | 92.7 ± 2.8 | 95.3 ± 3.0 | 96.3 ± 2.6 | 94.4 ± 3.2 | 95.4 ± 2.9 | **96.7** ± 2.2 |
| sonar | 72.6 ± 4.2 | 73.0 ± 6.4 | 72.6 ± 4.5 | 71.2 ± 4.2 | 72.7 ± 6.0 | **75.2** ± 4.8 |
| glass | 66.6 ± 2.9 | 66.1 ± 4.0 | 64.6 ± 3.2 | **67.4** ± 3.8 | 66.1 ± 4.2 | 66.2 ± 3.5 |
| newthyroid | 87.6 ± 3.5 | 88.7 ± 4.0 | **91.6** ± 3.2 | 89.9 ± 4.3 | 88.7 ± 4.0 | 90.7 ± 2.2 |
| spectfheart | 47.4 ± 2.3 | 41.9 ± 8.5 | 46.7 ± 6.9 | **49.1** ± 4.4 | 40.9 ± 7.3 | 46.1 ± 7.8 |
| heart | **77.3** ± 2.0 | 75.0 ± 2.6 | 75.7 ± 4.1 | 77.2 ± 3.9 | 74.4 ± 3.0 | 76.9 ± 2.3 |
| bupa | 54.1 ± 3.1 | 55.4 ± 3.4 | 53.9 ± 3.7 | **55.9** ± 4.1 | 55.4 ± 3.4 | 54.8 ± 3.5 |
| iono | 78.4 ± 2.6 | 77.7 ± 3.7 | 77.5 ± 3.8 | 78.2 ± 3.9 | 76.9 ± 3.9 | **79.9** ± 3.9 |
| libras | 68.3 ± 8.1 | 76.7 ± 8.5 | 69.7 ± 13.8 | 69.2 ± 11.0 | 69.0 ± 14.7 | **78.1** ± 9.4 |
| wdbc | 93.4 ± 1.3 | 93.5 ± 2.2 | 94.0 ± 1.4 | 93.6 ± 1.5 | 93.0 ± 2.2 | **94.8** ± 1.3 |
| balance | 87.4 ± 1.9 | 89.6 ± 1.5 | **92.1** ± 1.4 | 89.8 ± 1.9 | 89.5 ± 1.5 | 90.5 ± 1.4 |
| australian | 80.3 ± 1.6 | 80.7 ± 3.2 | **82.4** ± 1.5 | 81.1 ± 1.8 | 81.1 ± 3.2 | 82.1 ± 1.6 |
| pima | 60.1 ± 2.6 | 60.1 ± 2.1 | 60.8 ± 2.2 | 60.3 ± 2.8 | 60.1 ± 2.1 | **61.2** ± 2.0 |
| vehicle | 80.6 ± 2.1 | 92.0 ± 1.6 | 89.9 ± 3.1 | 89.5 ± 2.1 | **92.3** ± 1.5 | 91.1 ± 1.4 |
| german | 46.3 ± 2.2 | 45.4 ± 3.5 | 46.4 ± 1.8 | 46.0 ± 2.3 | 45.1 ± 3.4 | **47.1** ± 2.0 |
| yeast | 65.9 ± 2.9 | 67.1 ± 3.7 | **70.4** ± 2.8 | 68.4 ± 2.3 | 68.2 ± 3.7 | 70.4 ± 3.0 |
| segmentation | 82.0 ± 1.9 | 83.8 ± 2.9 | 81.6 ± 2.2 | 81.5 ± 1.8 | **84.8** ± 3.2 | 84.8 ± 2.2 |
| splice | 74.9 ± 0.9 | 86.3 ± 0.8 | 79.6 ± 1.2 | 76.4 ± 1.3 | **87.9** ± 1.1 | 87.2 ± 0.6 |
| abalone | **32.3** ± 0.7 | 31.4 ± 1.7 | 31.9 ± 0.8 | 31.7 ± 1.1 | 31.5 ± 1.2 | 32.3 ± 1.0 |
| spambase | 85.9 ± 0.7 | 88.5 ± 0.6 | 87.4 ± 0.9 | 87.2 ± 0.8 | 88.8 ± 0.8 | **89.4** ± 0.8 |
| pageblocks | 62.0 ± 2.9 | 61.5 ± 4.1 | 55.5 ± 4.0 | 61.5 ± 3.5 | 61.0 ± 4.1 | **62.5** ± 3.5 |
| Mean | 71.1 ± 2.7 | 72.5 ± 3.6 | 72.2 ± 3.5 | 72.2 ± 3.3 | 72.1 ± 3.9 | **73.9** ± 3.1 |
| Average Rank | 4.26 | 3.91 | 3.39 | 3.39 | 4.17 | **1.87** |

which modifies only locally the decision boundaries. However, these ellipsoids have no impact on regions of the feature space where there is no positive example. Instead, the SMOTE algorithm has this capacity to expand the decision boundaries by placing new synthetic examples in areas not covered by the ellipses.

**Third experiment—increasing the imbalance** We now aim at showing the efficiency of our method by artificially increasing and decreasing the imbalance. For a given dataset, we create a maximum of 10 synthetic variants where the percentage of minority examples is in $\{50\%, 40\%, 30\%, 20\%, 10\%, 5\%, 4\%, 3\%, 2\%, 1\%\}$. To create a synthetic variant of a dataset with a percentage of minority examples higher than in the original dataset, we apply a random under sampling of the majority class until the desired percentage is reached. Similarly, to create a synthetic variant with a smaller percentage of minority examples, we apply a random under sampling of the minority class. We create the synthetic variant of the dataset only if it contains at least 20 minority examples. For example, for the dataset SPECTFHEART, we cannot go under 10% of minority examples. Due to the small number of minority examples present in the more imbalanced synthetic variants of the datasets, we split them into 50% training and 50% test

*Table 2.4: Same experiment as in Table 2.2 after having applied a Random Under Sampling of the negative examples until $m^-$ reaches $m^+$.*

| Dataset | Euclidean | LMNN | ITML | GMML | IMLS | IML |
|---|---|---|---|---|---|---|
| hayes | 63.4 ± 9.0 | **67.7** ± 7.4 | 64.7 ± 6.7 | 66.4 ± 8.1 | 67.7 ± 7.4 | 66.0 ± 7.6 |
| wine | 91.2 ± 2.6 | 94.1 ± 2.8 | **94.2** ± 3.8 | 92.7 ± 3.7 | 94.1 ± 2.9 | 93.9 ± 3.2 |
| sonar | 70.4 ± 5.2 | 73.1 ± 6.3 | 70.2 ± 5.7 | 69.9 ± 5.5 | 71.1 ± 9.2 | **74.4** ± 4.8 |
| glass | 64.6 ± 3.5 | 63.2 ± 4.5 | 61.1 ± 4.9 | **64.6** ± 3.1 | 62.7 ± 5.0 | 64.5 ± 4.7 |
| newthyroid | 86.6 ± 4.6 | 91.4 ± 5.0 | 91.1 ± 4.9 | 90.6 ± 3.3 | 91.4 ± 5.0 | **92.3** ± 2.6 |
| spectfheart | 44.2 ± 3.9 | 42.6 ± 8.0 | 46.7 ± 4.6 | 45.9 ± 5.6 | 42.6 ± 8.0 | **48.8** ± 6.3 |
| heart | **77.4** ± 2.0 | 75.8 ± 3.3 | 76.6 ± 2.5 | 77.3 ± 1.9 | 75.5 ± 3.3 | 77.1 ± 2.1 |
| bupa | 53.8 ± 4.1 | 54.1 ± 4.8 | 54.6 ± 4.1 | **55.7** ± 3.6 | 54.1 ± 4.8 | 55.0 ± 3.2 |
| iono | 73.1 ± 5.2 | 73.3 ± 4.1 | 75.4 ± 3.4 | 74.7 ± 3.2 | 73.3 ± 4.1 | **77.3** ± 3.0 |
| libras | 34.3 ± 10.6 | 35.6 ± 10.9 | 38.2 ± 12.2 | 36.4 ± 12.6 | 35.6 ± 10.9 | **39.3** ± 13.8 |
| wdbc | 93.7 ± 1.2 | 92.9 ± 1.6 | 93.6 ± 1.8 | 93.2 ± 2.1 | 92.3 ± 2.5 | **94.7** ± 1.4 |
| balance | 87.5 ± 1.5 | 90.1 ± 1.3 | **92.8** ± 1.5 | 90.1 ± 1.7 | 90.2 ± 1.3 | 90.7 ± 1.4 |
| australian | 80.4 ± 1.7 | 81.7 ± 2.2 | 82.2 ± 1.6 | 81.5 ± 2.3 | 81.7 ± 2.2 | **82.5** ± 2.2 |
| pima | 60.8 ± 2.7 | 60.5 ± 2.1 | **62.2** ± 1.7 | 60.8 ± 2.4 | 60.4 ± 2.1 | 61.2 ± 2.5 |
| vehicle | 74.0 ± 3.1 | 89.7 ± 1.6 | 87.7 ± 2.6 | 85.5 ± 3.1 | **89.7** ± 2.0 | 89.3 ± 1.9 |
| german | 46.7 ± 1.6 | 46.9 ± 2.5 | 47.3 ± 2.3 | 47.5 ± 1.6 | 46.2 ± 2.0 | **48.0** ± 1.8 |
| yeast | 57.2 ± 4.5 | 60.8 ± 4.6 | 60.9 ± 3.8 | 59.7 ± 3.8 | 61.2 ± 4.9 | **61.9** ± 3.9 |
| segmentation | 64.6 ± 3.1 | 70.4 ± 2.4 | 65.7 ± 3.4 | 64.3 ± 2.9 | 72.4 ± 2.9 | **74.2** ± 1.8 |
| splice | 75.9 ± 0.7 | 86.5 ± 0.6 | 79.5 ± 1.6 | 76.2 ± 1.2 | **87.9** ± 0.9 | 87.2 ± 0.6 |
| abalone | **32.8** ± 1.1 | 32.5 ± 1.4 | 32.5 ± 1.3 | 31.6 ± 1.0 | 32.2 ± 1.3 | 32.6 ± 1.4 |
| spambase | 85.0 ± 1.0 | 88.1 ± 1.1 | 86.8 ± 1.2 | 86.2 ± 1.2 | 88.4 ± 0.7 | **88.7** ± 0.8 |
| pageblocks | 46.8 ± 3.7 | **50.2** ± 4.8 | 43.0 ± 5.2 | 48.3 ± 4.5 | 49.6 ± 5.6 | 49.1 ± 4.0 |
| Mean | 66.6±3.5 | 69.1±3.8 | 68.5±3.7 | 68.1±3.6 | 69.1±4.0 | **70.4**±3.4 |
| Average Rank | 4.83 | 3.65 | 3.30 | 3.96 | 3.43 | **1.83** |

examples. We report the mean results over 20 iterations where at each iteration we recompute the synthetic variants of the dataset and the train/test splits.

The results for the SPECTFHEART dataset (already used in the introduction of this chapter) are reported in Figure 2.4. We see that like the other algorithms, **IML** shows the same drop of F1-measure when increasing the imbalance, which shows the difficulty of learning from highly imbalanced data. However, it is important to notice that the drop of performances of **IML** is the smallest among all algorithms.

To confirm the efficiency of **IML** when facing imbalance data on a wide range of datasets, we present in Figure 2.5 the results of the same experiments by averaging the results over the 22 datasets. We observe the same behavior as for the SPECTFHEART dataset. Again, it is worth noticing that **IML** is always more robust while facing imbalanced classes.

**Fourth experiment—analyzing why IML is better than the other metric learning algorithms on imbalanced data** When we described **IML** in Section 2.3, we presented two strategies to deal with the imbalance. The first one, which is already used by some existing metric learning methods, is a selection of the similar and dissimilar pairs based on the nearest

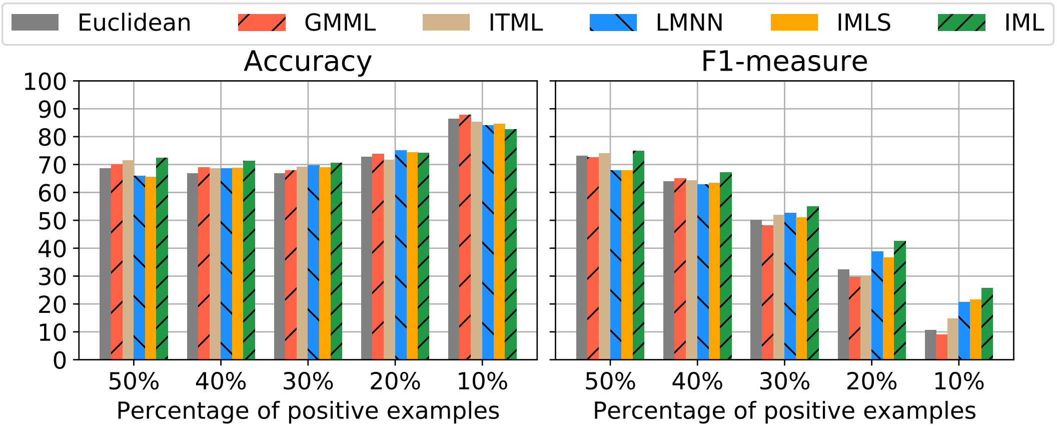


*Figure 2.4: Mean Accuracy and F1-measure over* 20 *splits on the* SPECTFHEART *dataset by artificially increasing the imbalance. We compare state-of-the-art metric learning algorithms with our proposed method* **IML***.*

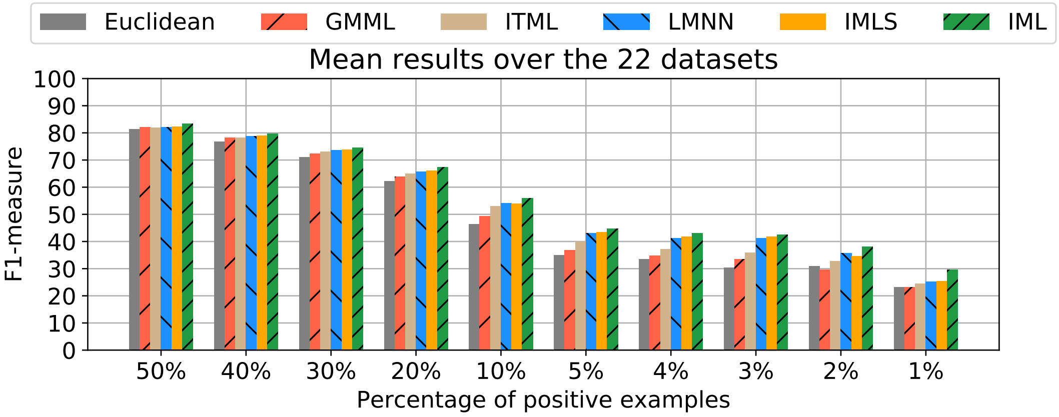


*Figure 2.5: Mean F1-measure over* 20 *splits and over the* 22 *datasets by artificially increasing the imbalance. We compare state-of-the-art metric learning algorithms with our proposed method* **IML***.*

neighbor rule. We proposed in this chapter a second method consisting in weighting differently the set of pairs based on the labels of the two examples composing the pairs. To see the impact of these two strategies, we compare in this last experiment **IML** with two variants.

The variant called **ML2** considers the loss of **IML** without the re-weighting of the set of pairs. Its loss is defined as follows:

$$\min_{\mathbf{M}\succeq 0} F(\mathbf{M}) = \frac{1}{m^2}\Bigg( \sum_{(\mathbf{z},\mathbf{z}')\in Sim^+} \ell_1(\mathbf{M},\mathbf{z},\mathbf{z}') + \sum_{(\mathbf{z},\mathbf{z}')\in Sim^-} \ell_1(\mathbf{M},\mathbf{z},\mathbf{z}') + \sum_{(\mathbf{z},\mathbf{z}')\in Dis^+} \ell_2(\mathbf{M},\mathbf{z},\mathbf{z}') + \sum_{(\mathbf{z},\mathbf{z}')\in Dis^-} \ell_2(\mathbf{M},\mathbf{z},\mathbf{z}') \Bigg) + \lambda\|\mathbf{M}-\mathbf{I}\|_F^2, \qquad (2.24)$$

where the difference with Equation (2.3) is that we no longer multiply each of the four sets respectively by $a$, $(1-a)$, $b$ and $(1-b)$.

The variant called **ML1** considers the same loss as **ML2**, but we select randomly the pairs of examples. In order to use the same number of pairs in **ML1** and **IML**, we draw randomly $2mk$ pairs for **ML1** since **IML** considers $k$ similar pairs and $k$ dissimilar pairs per training example.

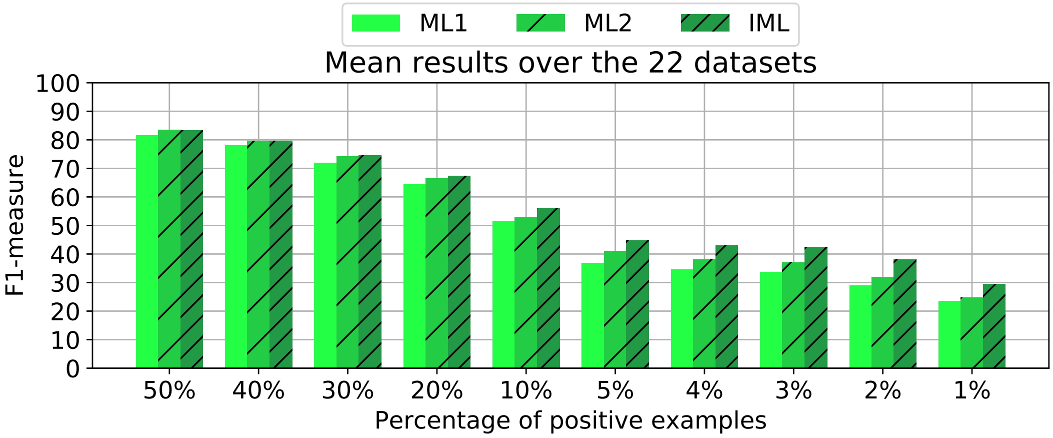


*Figure 2.6: Mean F1-measure over* 20 *splits and over the* 22 *datasets by artificially increasing the imbalance. We compare two variants of* **IML**. *The variant* **ML2** *removes one component that allows* **IML** *to deal with the imbalance: the re-weighting of the pairs.* **ML1** *removes in addition another component to deal with the imbalance: instead of selecting the similar and dissimilar pairs with the nearest neighbor rule, they are selected randomly by* **ML1**.

The results of this experiment are reported in Figure 2.6. When the classes are balanced with 50% of minority examples, we observe that **IML** and its two variants present more or less the same performances. When increasing the imbalance, as expected, **IML** tends to be better than **ML2**, this latter being better than **ML1**. This shows that our two strategies to deal with imbalanced data do not degrade the results on balanced data and that they are complementary to improve the performances in an imbalanced setting.

**Fifth experiment—incorporating our two strategies to deal with the imbalance in an existing metric learning algorithm** We saw in the previous experiment that our two strategies to deal with the imbalance improve the performances of our metric learning algorithm when the datasets are highly imbalanced without affecting the results when the data is balanced. We propose here to investigate if this improvement can also be observed by incorporating our strategies to an existing metric learning algorithm. We do not consider **LMNN** and **IMLS** because they make use of triplet constraints while our re-weighting is designed for pairwise constraints. We also do not consider **ITML** because our re-weighting sometimes causes the metric learned to be no longer PSD. Thus, we only consider **GMML** for this experiment as it is the most suited metric learning algorithm to adapt with our two strategies.

Our two modifications of **GMML** are the following: *(i)* instead of selecting randomly the pairs, we select them using the 3 nearest neighbors; *(ii)* instead of having the same weight for every pair, we assign to the pairs where the first example is a positive example a weight equal to the number of negative examples $m^-$, and to the other pairs a weight equal to the number of positive examples $m^+$. We call our modification of this algorithm **IGMML**.

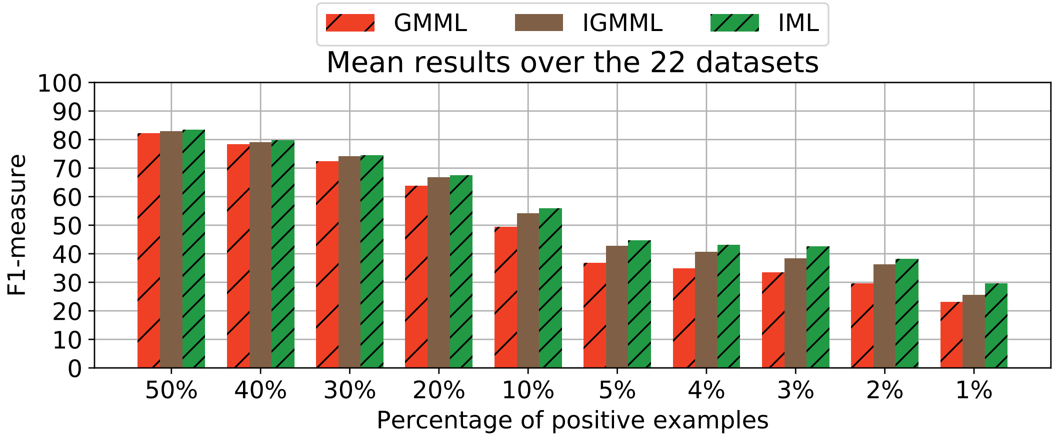


*Figure 2.7: Mean F1-measure over* 20 *splits and over the* 22 *datasets by artificially increasing the imbalance. We compare the existing metric learning algorithm* ***GMML*** *with a variant (**IGMML**) that incorporates our two strategies to deal with the imbalance.*

We report the results of this experiment in Figure 2.7. As already noticed, we observe that the algorithms present the same performances when the data is balanced. On the other hand, the variant built upon our two strategies tend to improve the results when increasing the imbalance. This shows that our contributions are not specific to our proposed metric learning method **IML** and can also be adapted to existing metric learning methods. However, it is worth noticing that plugged into **IML** allows us to get better results. Even if the two strategies are not specific to **IML**, they work better for our method than for **IGMML** when facing a percentage of positive examples lower or equal to 10%.

## 2.6 Conclusion and perspectives

In this chapter, we revisit the classic formulation of metric learning algorithms that learn a Mahalanobis metric in the light of imbalanced data issues. Our method resorts to two complementary strategies to deal with the imbalance. First, unlike the state of the art methods that do not make any distinction between the pairs, we propose to decompose the usual loss with respect to the different possible labels involved in the pairs of examples. This decomposition allows us to assign specific weights to each type of pairs in order to improve the performance on the minority class. We derive a generalization bound specific to the imbalanced setting showing a convergence term depending on the class imbalance and illustrating the hardness of learning from imbalanced data. Our experimental evaluation shows that we are able to obtain better results than state of the art metric learning algorithms in terms of F1-measure over balanced and imbalanced datasets. Last but not least, artificially increasing the imbalance in the datasets shows that our two strategies to deal with the imbalance are complementary while easily adaptable to existing metric learning algorithms.

We believe that our work gives rise to exciting perspectives when facing imbalanced data. Among them, we want to study how our algorithm could be adapted to learn non-linear metrics. From an algorithmic point of view, we would like to extend our method by deriving a closed form solution in a similar way as done by Zadeh et al. [2016] to drastically reduce the computation time while maintaining good performances.

# Chapter 3

# Ensemble Learning with Random Fourier Features and Boosting

**This chapter is based on the following publications**

Léo Gautheron, Pascal Germain, Amaury Habrard, Guillaume Metzler, Emilie Morvant, Marc Sebban and Valentina Zantedeschi. Landmark-based Ensemble Learning with Random Fourier Features and Gradient Boosting. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2020 [Gautheron et al., 2020b].

Léo Gautheron, Pascal Germain, Amaury Habrard, Guillaume Metzler, Emilie Morvant, Marc Sebban and Valentina Zantedeschi. Apprentissage d'ensemble basé sur des points de repère avec des caractéristiques de Fourier aléatoires et un renforcement du gradient. In *Conférence sur l'Apprentissage automatique (CAp)*, 2020 [Gautheron et al., 2020a].

Léo Gautheron, Pascal Germain, Amaury Habrard, Gaël Letarte, Emilie Morvant, Marc Sebban and Valentina Zantedeschi. Revisite des "random Fourier features" basée sur l'apprentissage PAC-Bayésien via des points d'intérêts. In *Conférence sur l'Apprentissage automatique (CAp)*, 2019, Toulouse, France [Gautheron et al., 2019a].

**Abstract**

This chapter jointly leverages two state-of-the-art learning strategies—gradient boosting (GB) and kernel Random Fourier Features (RFF)—to address the problem of kernel learning. Our study builds on a recent result showing that one can learn a distribution over the RFF to produce a new kernel suited for the task at hand. For learning this distribution, we exploit a GB scheme expressed as ensembles of RFF weak learners, each of them being a kernel function designed to fit the residual. Unlike Multiple Kernel Learning techniques that make use of a pre-computed dictionary of kernel functions to select from, at each iteration we fit a kernel by approximating it from the training data as a weighted sum of RFF. This strategy allows one to build a classifier based on a small ensemble of learned kernel “landmarks” better suited for the underlying application. We conduct a thorough experimental analysis to highlight the advantages of our method compared to both boosting-based and kernel-learning state-of-the-art methods. Our results show that our method, thanks to an approximation of non-linear kernels, is able to learn quickly complex decision boundaries that generalizes well with a small number of training examples.

## 3.1 Introduction

Kernel methods (recalled in Section 1.2.2) are among the most popular approaches in machine learning due to their capability to address non-linear problems, their robustness and their simplicity. However, they exhibit two main flaws in terms of memory usage and time complexity. Landmark-based kernel approaches [Balcan et al., 2008] can be used to drastically reduce the number of instances involved in the comparisons, but they heavily depend on the choice and the parameterization of the kernel. Multiple Kernel Learning [Wu et al., 2017] and Matching Pursuit methods [Vincent and Bengio, 2002] can provide alternative solutions to this problem but they require the use of a pre-defined dictionary of base functions. Another strategy to improve the scalability of kernel methods is to use approximation techniques such as the Nyström [Drineas and Mahoney, 2005] or Random Fourier Features (RFF) [Rahimi and Recht, 2008] (recalled in Section 1.3.2). The latter is probably the most used thanks to its simplicity and ease of computation. It allows the approximation of any shift-invariant kernel based on the Fourier transform of the kernel. Several works have extended this technique by allowing one to adapt the RFF approximation directly from the training data [Agrawal et al., 2019, Letarte et al., 2019, Sinha and Duchi, 2016]. Among them, the recent work of Letarte et al. [2019] introduces a method to obtain a weighting distribution over the random features by a single pass over them. This strategy is derived from a statistical learning analysis, starting from the observation that each random feature can be interpreted as a weak hypothesis in the form of trigonometric functions obtained by the Fourier decomposition. However, in practice, this method requires the use of a fixed set of landmarks selected beforehand and independently from the task before being able to learn the representation in a second step. This leads to three important limitations: *(i)* the need for a heuristic strategy for selecting relevant landmarks, *(ii)* these latter and the associated representation might not be adapted for the underlying task, and *(iii)* the number of landmarks might not be minimal *w.r.t.* that task, inducing higher computational and memory costs.

We propose in this chapter to tackle these issues with a gradient boosting approach [Friedman, 2001] based on the boosting framework recalled in Section 1.2.5. Our aim is to learn iteratively the classifier and a compact and efficient representation at the same time. Our greedy optimization method is similar to the one from Oglic and Gärtner [2016], which at each iteration of the functional gradient descent [Mason et al., 1999] refines the representation by adding the base function minimizing a residual-based loss function. But unlike our approach, their method does not allow to learn a classifier at the same time. Instead, we propose to jointly optimize the classifier and the base functions in the form of kernels by leveraging both gradient boosting and RFF. Interestingly, we further show that we can benefit from a significant performance boost by *(i)* considering each weak learner as a single trigonometric feature, and *(ii)* learning the random part of the RFF. This way to proceed allows us to learn well even from small datasets.

**Organization of the chapter.** Section 3.2 describes the notations and the necessary background knowledge. We present our method in Section 3.4 as well as two efficient refinements before presenting an extensive experimental study in Section 3.5, comparing our strategy with boosting-based and kernel learning methods.

## 3.2 Notations and related work

In this chapter, we consider binary classification tasks as introduced in Chapter 1. We focus on kernel-based algorithms that rely on pre-defined kernel functions $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ assessing the similarity between any two points of the input space. These methods present a good performance when the parameters of the kernels are learned and the chosen kernels are able to fit the distribution of the data, as shown in Figure 1.7. However, selecting the right kernel and tuning its parameters is computationally expensive, in general. To reduce this overhead, one can resort to Multiple Kernel Learning techniques [Wu et al., 2017] which boils down to selecting the combination of kernels that fits the best the training data: a dictionary of $T$ base functions $\{k^t\}_{t=1}^T$ is composed of various kernels associated with some fixed parameters, and a combination is learned, defined as

$$H(\boldsymbol{x}, \boldsymbol{x}') \;=\; \sum_{t=1}^{T} \alpha^t \, k^t(\boldsymbol{x}, \boldsymbol{x}'), \tag{3.1}$$

with $\alpha^t \in \mathbb{R}$ the weight of the kernel $k^t(\boldsymbol{x}, \boldsymbol{x}')$. As shown in Section 3.4, our main contribution is to address this issue of optimizing a linear combination of kernels by leveraging RFF and gradient boosting (we recall basics on it in Section 3.4.1). To avoid the dictionary of kernel functions in Equation (3.1) from being pre-computed, we propose a method inspired from Letarte et al. [2019] to learn a set of approximations of kernels tailored to the underlying classification task. Unlike Letarte et al. [2019], we learn such functions so that the representation and the classifier are jointly optimized. We consider landmark-based shift-invariant kernels relying on the value $\boldsymbol{\delta} = \boldsymbol{x}^t - \boldsymbol{x} \in \mathbb{R}^d$ and usually denoted by abuse of notation by

$$k(\boldsymbol{\delta}) = k(\boldsymbol{x}^t - \boldsymbol{x}) = k(\boldsymbol{x}^t, \boldsymbol{x}),$$

where $\boldsymbol{x}^t \in \mathbb{R}^d$ is a point—called landmark—lying on the input space which all the instances are compared to, and that strongly characterizes the kernel.

At each iteration of our gradient boosting procedure, we optimize not only this landmark but also the kernel function itself, exploiting the flexibility of the framework provided by Letarte et al. [2019]. We write the kernel as a sum of RFF [Rahimi and Recht, 2008] and we learn a posterior distribution over them. We achieve this by studying the generalization capabilities of the so-defined functions through the lens of the PAC-Bayesian recalled in Section 1.1.5. This theoretical analysis ultimately allows us to derive a closed-form solution of the posterior distribution $q^t$ (over the RFF at a given iteration $t$), which is guaranteed to minimize a generalization bound on the loss of the kernel. In the following section, we recall the framework of Letarte et al. and adapt it to our scenario.

## 3.3 Pseudo-bayesian kernel learning with RFF

The kernel learning method proposed by Letarte et al. [2019] builds on the RFF approximation proposed by Rahimi and Recht [2008] and recalled in Section 1.3.2. Instead of drawing RFF for approximating a known kernel, Letarte et al. propose to learn a new one by deriving a posterior distribution $q^t$ for a given landmark point in $\{\boldsymbol{x}^t\}_{t=1}^T$:

$$k_{q^t}(\boldsymbol{x}^t - \boldsymbol{x}) = \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim q^t} \cos\left(\boldsymbol{\omega} \cdot (\boldsymbol{x}^t - \boldsymbol{x})\right).$$

Their kernel learning approach first considers $p$ the Fourier transform of a given kernel $k$ as a prior distribution. Then, they aim at learning the posterior distribution $q^t$ by minimizing a PAC-Bayesian generalization bound on the expected value of a loss between the landmark $(\boldsymbol{x}^t, y^t)$ and any point $(\boldsymbol{x}, y) \sim \mathcal{D}$. Following the RFF framework, this kernel can be approximated in practice by drawing $K$ vectors $\boldsymbol{\omega}$ according to $p$ and then re-weighting each random feature according to $q^t$ by computing

$$k_{\boldsymbol{q}^t}(\boldsymbol{x}^t - \boldsymbol{x}) = \sum_{j=1}^{K} q_j^t \cos\left(\boldsymbol{\omega}_j \cdot (\boldsymbol{x}^t - \boldsymbol{x})\right), \tag{3.2}$$

where $\boldsymbol{q}^t$ is the empirical counterpart of the distribution $q^t$ defined such that

$$\forall j \in \{1, \dots, K\},\ 0 \le q_j^t \le 1, \quad \text{and} \quad \sum_{j=1}^{K} q_j^t = 1.$$

Let $(\boldsymbol{x}^t, y^t)$ be an example called landmark and $k_{q^t}$ the kernel built using this landmark, then the true risk $R(k_{q^t})$ and empirical risk $\widehat{R}(k_{q^t})$ of the kernel are respectively defined as

$$R(k_{q^t}) = \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{D}} \ell\left(k_{q^t}(\boldsymbol{x}^t - \boldsymbol{x})\right), \quad \text{and} \quad \widehat{R}(k_{q^t}) = \frac{1}{m-1} \sum_{j=1, j\neq t}^{m} \ell\left(k_{q^t}(\boldsymbol{x}^t - \boldsymbol{x}_j)\right).$$

Using the PAC-Bayesian theory, they obtain the following theorem, under the linear loss $\ell\left(k_{q^t}(\boldsymbol{x}^t - \boldsymbol{x})\right) = \frac{1}{2} - \frac{1}{2} y^t y k_{q^t}(\boldsymbol{x}^t - \boldsymbol{x})$, by expressing the loss as

$$R(k_{q^t}) \ = \ R\left(\mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim p} h_{\boldsymbol{\omega}}^t\right)$$

$$= \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim p} R(h^t_{\boldsymbol{\omega}}),$$

with $h^t_{\boldsymbol{\omega}}(\boldsymbol{x}) = \cos(\boldsymbol{\omega} \cdot (\boldsymbol{x}^t - \boldsymbol{x}))$.

**Theorem 5** (Theorem 1 from Letarte et al. [2019]). *For $s > 0$, $\forall i \in \{1, \ldots, m\}$, and a prior distribution $p$ over $\mathbb{R}^d$, with probability $1 - \delta$ over the random choice of $S \sim \mathcal{D}^m$, we have $\forall q \in \mathbb{R}^d$:*

$$R(k_q) \leq \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim p} \widehat{R}(h^t_{\boldsymbol{\omega}}) + \frac{1}{s}\left( KL(q\|p) + \frac{s^2}{2(m-1)} + \ln\frac{1}{\delta} \right),$$

*where $KL(q\|p) = \mathbb{E}_{\boldsymbol{\omega}\sim p} \frac{p(\boldsymbol{\omega})}{q(\boldsymbol{\omega})}$ is the Kullback-Leibler divergence between $q$ and $p$.*

We note that the result also stands for any $[0,1]$-valued convex loss $\ell$. Indeed, by Jensen's inequality, we have

$$R(k_{q^t}) = R\left( \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim p} h^t_{\boldsymbol{\omega}} \right) \leq \mathop{\mathbb{E}}_{\boldsymbol{\omega}\sim p} R(h^t_{\boldsymbol{\omega}}).$$

As described in Section 1.1.5, this bound is interesting because there exists a closed form solution minimizing it (see Germain et al. [2009] and Letarte et al. [2019]) and computed in this setting as

$$\forall j \in \{1, \ldots, K\}, \quad q^t_j = \frac{1}{Z^t} \exp\left( \frac{-\beta\sqrt{m}}{m} \sum_{i=1}^{m} \ell\left( h^t_{\boldsymbol{\omega}_j}(\boldsymbol{x}_i) \right) \right), \tag{3.3}$$

with $\beta \geq 0$ a parameter to tune and $Z^t$ a normalization constant such that $\sum_{j=1}^{K} q^t_j{=}1$.

The proposed method of Letarte et al. [2019] consists in a first step to learn a representation of the input space of $n_L$ features where each new feature $\forall t \in \{1, \ldots, n_L\}$ is computed using the kernel $k_{\boldsymbol{q}^t}$ computed according to Equations (3.2) and (3.3) with the landmark $(\boldsymbol{x}^t, y^t)$. To do so, they consider a set of $n_L$ landmarks $L = \left\{(\boldsymbol{x}^t, y^t)\right\}_{t=1}^{n_L}$ which they choose either as randomly from the training set, or as the centers of a clustering of the training set. Then, during a second step, a (linear) predictor can be learned from the new representation.

It is worth noticing that this kind of procedure exhibits two limitations. First, the model can be optimized only after having learned the representation. Second, the landmarks have to be fixed before learning the representation. Thus, the constructed representation is not guaranteed to be compact and relevant for the learning algorithm considered. To tackle these issues, we propose in the following a strategy that performs both steps at the same time through a gradient boosting process that allows to jointly learn the set of landmarks and the final predictor.

## 3.4 Gradient boosting random Fourier features

The approach we propose follows the widely used gradient boosting framework first introduced by Friedman [2001]. We briefly recall it below.

**Algorithm 3.1:** Gradient boosting [Friedman, 2001]

```
Inputs : Training set $S = \big\{(\boldsymbol{x}_i, y_i)\big\}_{i=1}^m$; Loss $\ell$; Number of iterations $T$
Output: sign $\Big(H^0(\boldsymbol{x}) + \sum_{t=1}^T \alpha^t h_{\mathbf{a}^t}(\boldsymbol{x})\Big)$
1: $\forall i \in \{1, \dots, m\} \quad H^0(\boldsymbol{x}_i) = \arg\min_\rho \sum_{i=1}^m \ell(y_i, \rho)$
2: for $t \in \{1, \dots, T\}$ do
3:     $\forall i \in \{1, \dots, m\} \quad \tilde{y}_i = -\frac{\partial \ell\big(y_i, H^{t-1}(\boldsymbol{x}_i)\big)}{\partial H^{t-1}(\boldsymbol{x}_i)}$
4:     $\mathbf{a}^t = \arg\min_{\mathbf{a}} \sum_{i=1}^m \big(\tilde{y}_i - h_{\mathbf{a}}(\boldsymbol{x}_i)\big)^2$
5:     $\alpha^t = \arg\min_\alpha \sum_{i=1}^m \ell\big(y_i, H^{t-1}(\boldsymbol{x}_i) + \alpha h_{\mathbf{a}^t}(\boldsymbol{x}_i)\big)$
6:     $\forall i \in \{1, \dots, m\} \quad H^t(\boldsymbol{x}_i) = H^{t-1}(\boldsymbol{x}_i) + \alpha^t h_{\mathbf{a}^t}(\boldsymbol{x}_i)$
7: end for
```

### 3.4.1 Gradient boosting in a nutshell

Gradient boosting is an ensemble method that aims at learning a weighted majority vote over an ensemble of $T$ weak predictors in a greedy way by learning one classifier per iteration. The final majority vote is of the form

$$\forall \boldsymbol{x} \in \mathbb{R}^d, \ \operatorname{sign}\left(H^0(\boldsymbol{x}) + \sum_{t=1}^T \alpha^t h_{\mathbf{a}^t}(\boldsymbol{x})\right),$$

where $H^0$ is an initial classifier fixed before the iterative process (usually set such that it returns the same value for every example), and $\alpha^t$ is the weight associated to the predictor $h_{\mathbf{a}^t}$ and is learned at the same time as the parameters $\mathbf{a}^t$ of that classifier. Given a differentiable loss $\ell$, the objective of the gradient boosting algorithm is to perform a gradient descent where the variable to be optimized is the ensemble and the function to be minimized is the empirical loss. The pseudo-code of gradient boosting is reported in Algorithm 3.1. First, the ensemble is constituted by only one predictor: the one that outputs a constant value minimizing the loss over the whole training set (line **1**). Then at each iteration, the algorithm computes for each training example the negative gradient of the loss (line **3**), also called the residual and denoted by $\tilde{y}_i$. The next step consists in optimizing the parameters of the predictor $h_{\mathbf{a}^t}$ that fits the best the residuals (line **4**), before learning the optimal step size $\alpha^t$ that minimizes the loss by adding $h_{\mathbf{a}^t}$, weighted by $\alpha^t$, to the current vote (line **5**). Finally, the model is updated by adding $\alpha^t h_{\mathbf{a}^t}(\cdot)$ (line **6**) to the vote.

### 3.4.2 Gradient boosting with random Fourier features

Our main contribution takes the form of a learning algorithm which jointly optimizes a compact representation of the data and the model. Our method, called **GBRFF1**, leverages both Gradient Boosting and RFF. We describe its pseudo-code in Algorithm 3.2 which follows the steps of Algorithm 3.1. The loss function $\ell$ at the core of our algorithm is the exponential loss

$$\ell\left(H^T\right) = \frac{1}{m}\sum_{i=1}^m \exp\Big(-y_i\, H^T(\boldsymbol{x}_i)\Big). \tag{3.4}$$

**Algorithm 3.2:** GBRFF1

```
**Inputs :** Training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$; Number of iterations $T$;
             $K$ number of random features; Parameters $\gamma$ and $\beta$
**Output:** sign $\Big(H^0(\boldsymbol{x}) + \sum_{t=1}^T \alpha^t \sum_{j=1}^K q_j^t \cos\big(\boldsymbol{\omega}_j^t \cdot (\boldsymbol{x}^t - \boldsymbol{x})\big)\Big)$
$\forall i \in \{1, \dots, m\} \quad H^0(\boldsymbol{x}_i) = \frac{1}{2} \ln \frac{1 + \frac{1}{m}\sum_{j=1}^m y_j}{1 - \frac{1}{m}\sum_{j=1}^m y_j}$
**for** $t \in \{1, \dots, T\}$ **do**
    $\forall i \in \{1, \dots, m\} \quad w_i = \exp(-y_i H^{t-1}(\boldsymbol{x}_i))$
    $\forall i \in \{1, \dots, m\} \quad \tilde{y}_i = y_i w_i$
    $\forall j \in \{1, \dots, K\}$, draw $\boldsymbol{\omega}_j^t \sim \mathcal{N}(0, 2\gamma)^d$
    $\boldsymbol{x}^t = \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^m \exp\Big(-\tilde{y}_i \frac{1}{K} \sum_{j=1}^K \cos(\boldsymbol{\omega}_j^t \cdot (\boldsymbol{x} - \boldsymbol{x}_i))\Big)$
    $\forall j \in \{1, \dots, K\} \ q_j^t = \frac{1}{Z^t} \exp\Bigg(\frac{-\beta\sqrt{m}}{m} \sum_{i=1}^m \exp\Big(-\tilde{y}_i \cos\big(\boldsymbol{\omega}_j^t \cdot (\boldsymbol{x}^t - \boldsymbol{x}_i)\big)\Big)\Bigg)$
    $\alpha^t = \frac{1}{2} \ln \frac{\sum_{i=1}^m \Big(1 + y_i \sum_{j=1}^K q_j^t \cos\big(\boldsymbol{\omega}_j^t \cdot (\boldsymbol{x}^t - \boldsymbol{x}_i)\big)\Big) w_i}{\sum_{i=1}^m \Big(1 - y_i \sum_{j=1}^K q_j^t \cos\big(\boldsymbol{\omega}_j^t \cdot (\boldsymbol{x}^t - \boldsymbol{x}_i)\big)\Big) w_i}$
    $\forall i \in \{1, \dots, m\} \ H^t(\boldsymbol{x}_i) = H^{t-1}(\boldsymbol{x}_i) + \alpha^t \sum_{j=1}^K q_j^t \cos\big(\boldsymbol{\omega}_j^t \cdot (\boldsymbol{x}^t - \boldsymbol{x}_i)\big)$
**end for**
```

We show in Figure 3.1 the intuition on why the exponential loss is more adapted to binary classification tasks compared to the usually used least square loss in Gradient Boosting algorithms [Friedman, 2001]. Given $\ell$ as the exponential loss, line **1** of Algorithm 3.1 amounts to setting the initial learner as

$$\forall i \in \{1, \dots, m\} \quad H^0(\boldsymbol{x}_i) = \frac{1}{2} \ln \frac{1 + \frac{1}{m}\sum_{j=1}^m y_j}{1 - \frac{1}{m}\sum_{j=1}^m y_j}. \tag{3.5}$$

The residuals of line **3** are defined as $\tilde{y}_i = -\dfrac{\partial \ell\big(y_i, H^{t-1}(\boldsymbol{x}_i)\big)}{\partial H^{t-1}(\boldsymbol{x}_i)} = y_i \exp\big(-y_i H^{t-1}(\boldsymbol{x}_i)\big)$. Line **4** of Algorithm 3.1 tends to learn a weak learner that outputs exactly the residuals' values by minimizing the squared loss; but, this is not well suited in our setting with the exponential loss (Equation (3.4)). To benefit from the exponential decrease of the loss, we are rather interested in weak learners that output predictions having a large absolute value and being of the same sign as the residuals. Thus, we aim at favoring parameter values minimizing the exponential loss between the residuals and the predictions of the weak learner as follows:

$$\mathbf{a}^t = \operatorname*{arg\,min}_{\mathbf{a}} \frac{1}{m} \sum_{i=1}^m \exp\big(-\tilde{y}_i h_{\mathbf{a}}(\boldsymbol{x}_i)\big). \tag{3.6}$$

Following the RFF principle, we can now define our weak learner as

$$h_{\mathbf{a}^t}(\boldsymbol{x}_i) = \sum_{j=1}^K q_j^t \cos\big(\boldsymbol{\omega}_j^t \cdot (\boldsymbol{x}^t - \boldsymbol{x}_i)\big), \tag{3.7}$$

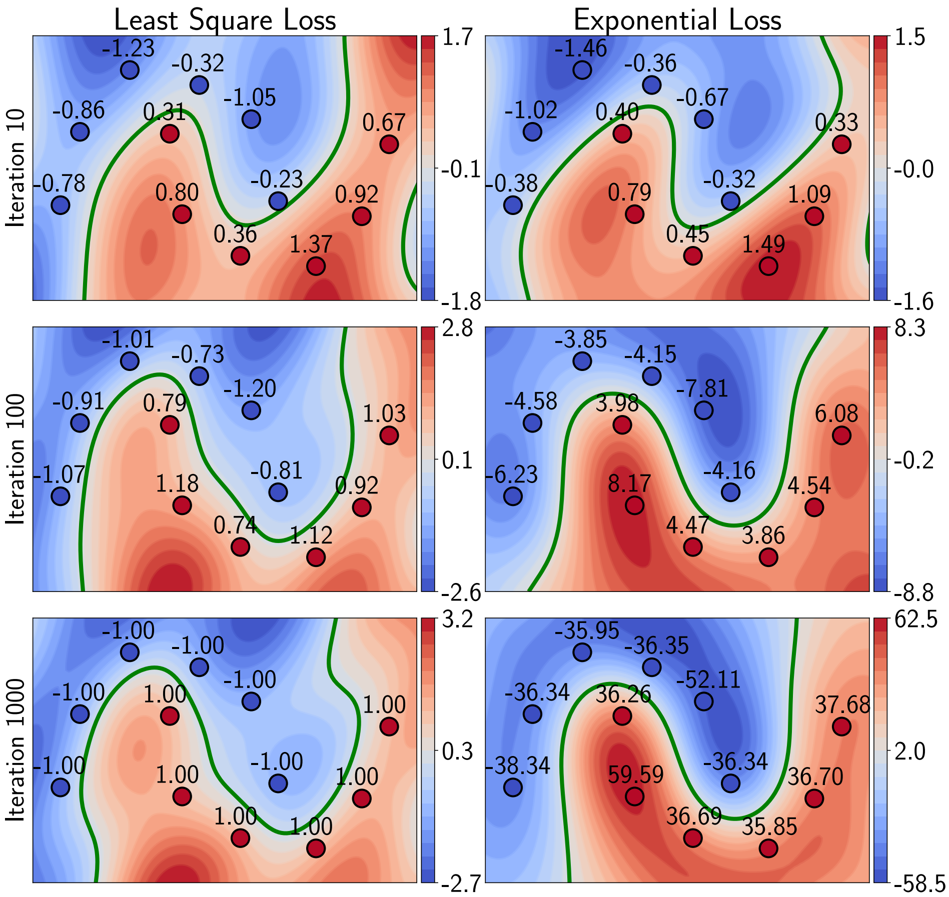


*Figure 3.1: Predicted values by our proposed method depending on the loss function used. The least square loss (left) is more adapted for regressions tasks as it encourages to predict exactly the true labels $-1$ and $+1$. On the right, the exponential loss is more adapted to binary classification tasks because it encourages to have predictions with the same sign as the true labels, resulting in a decision boundary with a larger margin to the examples.*

where its parameters are given by $\mathbf{a}^t{=}(\{\boldsymbol{\omega}_j^t\}_{j=1}^K, \boldsymbol{x}^t, \boldsymbol{q}^t)$. Instead of using a pre-defined set of landmarks [Letarte et al., 2019], we build this set iteratively, *i.e.*, we learn one landmark per iteration. To benefit from the closed form of Equation (3.3), we propose the following greedy approach to learn the parameters $\mathbf{a}^t$. At each iteration $t$, we draw $K$ vectors $\{\boldsymbol{\omega}_j^t\}_{j=1}^K{\sim}p^K$ with $p$ the Fourier transform of a given kernel (as defined in Equation (1.3)); then we look for the optimal landmark $\boldsymbol{x}^t$. Plugging Equation (3.7) into Equation (3.6) and assuming a uniform prior distribution over the random features, $\boldsymbol{x}^t$ is learned to minimize

$$\boldsymbol{x}^t = \underset{\boldsymbol{x}\in\mathbb{R}^d}{\arg\min}\ f(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^m \exp\Big(-\tilde{y}_i\frac{1}{K}\sum_{j=1}^K \cos(\boldsymbol{\omega}_j^t\cdot(\boldsymbol{x}-\boldsymbol{x}_i))\Big). \tag{3.8}$$

Even if this problem is non-convex due to the cosine function, we can still compute its derivative and perform a gradient descent to find a possible solution. The partial derivative of Equation (3.8) with respect to $\boldsymbol{x}$ is given by

$$\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x})=\frac{1}{Km}\sum_{i=1}^{m}\left[\frac{\tilde{y}_i}{K}\sum_{j=1}^{K}\sin(\boldsymbol{\omega}_j^t\cdot(\boldsymbol{x}-\boldsymbol{x}_i))\right]\exp\left[-\frac{\tilde{y}_i}{K}\sum_{j=1}^{K}\cos(\boldsymbol{\omega}_j^t\cdot(\boldsymbol{x}-\boldsymbol{x}_i))\right]\sum_{j=1}^{K}\boldsymbol{\omega}_j^t.$$

According to Letarte et al. [2019], given the landmark $\boldsymbol{x}^t$ found by gradient descent, we can now compute the weights of the random features $\boldsymbol{q}^t$ as

$$\forall j\in\{1,\dots,K\},\quad q_j^t=\frac{1}{Z^t}\exp\left[\frac{-\beta\sqrt{m}}{m}\sum_{i=1}^{m}\exp\Big(-\tilde{y}_i\cos\big(\boldsymbol{\omega}_j^t\cdot(\boldsymbol{x}^t-\boldsymbol{x}_i)\big)\Big)\right],\tag{3.9}$$

with $\beta\geq 0$ a parameter to tune and $Z^t$ the normalization constant.

The last step concerns the step size $\alpha^t$. It is computed so as to minimize the combination of the current model $H^{t-1}$ with the weak learner $h^t$, *i.e.*,

$$\begin{aligned}\alpha^t=&\arg\min_{\alpha}\sum_{i=1}^{m}\exp\left[-y_i(H^{t-1}(\boldsymbol{x}_i)+\alpha h^t(\boldsymbol{x}_i))\right]\\=&\arg\min_{\alpha}\sum_{i=1}^{m}w_i\exp\left[-y_i\alpha h^t(\boldsymbol{x}_i)\right],\end{aligned}$$

where $w_i=\exp(-y_iH^{t-1}(\boldsymbol{x}_i))$. In order to have a closed-form solution of $\alpha$, we use the convexity of the above quantity and the fact that $h^t(\boldsymbol{x}_i)\in[-1,1]$ to bound the loss function to optimize. Indeed, we get

$$\sum_{i=1}^{m}w_i\exp\big(-y_i\alpha h^t(\boldsymbol{x}_i)\big)\leq\sum_{i=1}^{m}\left[\frac{1-y_ih^t(\boldsymbol{x}_i)}{2}\right]w_i\exp(\alpha)+\sum_{i=1}^{m}\left[\frac{1+y_ih^t(\boldsymbol{x}_i)}{2}\right]w_i\exp(-\alpha).$$

This upper bound is strictly convex. Its minimum $\alpha^t$ can be found by setting to 0 the derivative *w.r.t.* $\alpha$ of the right-hand side of the previous equation. We get

$$\begin{aligned}&\sum_{i=1}^{m}\left(\frac{1-y_ih^t(\boldsymbol{x}_i)}{2}\right)w_i\ \exp(\alpha)\\=&\sum_{i=1}^{m}\left(\frac{1+y_ih^t(\boldsymbol{x}_i)}{2}\right)w_i\ \exp(-\alpha),\end{aligned}$$

for which the solution is given by $\alpha^t=\dfrac{1}{2}\ln\left(\dfrac{\sum_{i=1}^{m}(1-y_ih^t(\boldsymbol{x}_i))w_i}{\sum_{i=1}^{m}(1+y_ih^t(\boldsymbol{x}_i))w_i}\right)$.

The same derivation can be used to find the initial predictor $H^0$.

As usually done in the RFF literature [Agrawal et al., 2019, Rahimi and Recht, 2008, Sinha and Duchi, 2016] we use the RBF kernel defined as $k_\gamma(\boldsymbol{x},\boldsymbol{x}')=\exp(-\gamma\|\boldsymbol{x}-\boldsymbol{x}'\|^2)$ with as Fourier transform vectors of $d$ numbers each drawn from the normal law with zero mean and variance $2\gamma$ that we denote $\mathcal{N}(0,2\gamma)^d$.

### 3.4.3 Refining GBRFF1

In **GBRFF1**, the number of random features $K$ used at each iteration has a direct impact on the computation time of the algorithm. Moreover $\boldsymbol{\omega}^t$ is drawn according to the Fourier

**Algorithm 3.3:** GBRFF2

```
Inputs : Training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$; Number of iterations $T$;
         Parameters $\gamma$ and $\lambda$
Output: sign $\Big(H^0(\boldsymbol{x}) + \sum_{t=1}^T \alpha^t \cos\big(\boldsymbol{\omega}^t \cdot \boldsymbol{x} - b^t\big)\Big)$
1: $\forall i \in \{1, \dots, m\} \quad H^0(\boldsymbol{x}_i) = \frac{1}{2} \ln \frac{1+\frac{1}{m}\sum_{j=1}^m y_j}{1-\frac{1}{m}\sum_{j=1}^m y_j}$
2: for $t \in \{1, \dots, T\}$ do
3:     $\forall i \in \{1, \dots, m\} \quad w_i = \exp(-y_i H^{t-1}(\boldsymbol{x}_i))$
4:     $\forall i \in \{1, \dots, m\} \quad \tilde{y}_i = y_i w_i$
5:     Draw $\boldsymbol{\omega} \sim \mathcal{N}(0, 2\gamma)^d$
6:     $b^t = \underset{b \in [-\pi, \pi]}{\arg\min} \frac{1}{m} \sum_{i=1}^m \exp\Big( - \tilde{y}_i \cos\Big(\boldsymbol{\omega} \cdot \boldsymbol{x}_i - b\Big)\Big)$
7:     $\boldsymbol{\omega}^t = \underset{\boldsymbol{\omega} \in \mathbb{R}^d}{\arg\min} \lambda \|\boldsymbol{\omega}\|_2^2 + \frac{1}{m} \sum_{i=1}^m \exp\Big( - \tilde{y}_i \cos\Big(\boldsymbol{\omega} \cdot \boldsymbol{x}_i - b^t\Big)\Big)$.
8:     $\alpha^t = \frac{1}{2} \ln \frac{\sum_{i=1}^m \Big(1 + y_i \cos\Big(\boldsymbol{\omega}^t \cdot \boldsymbol{x}_i - b^t\Big)\Big) w_i}{\sum_{i=1}^m \Big(1 - y_i \cos\Big(\boldsymbol{\omega}^t \cdot \boldsymbol{x}_i - b^t\Big)\Big) w_i}$
9:     $\forall i \in \{1, \dots, m\} \; H^t(\boldsymbol{x}_i) = H^{t-1}(\boldsymbol{x}_i) + \alpha^t \cos\big(\boldsymbol{\omega}^t \cdot \boldsymbol{x}_i - b^t\big)$
10: end for
```

transform of the RBF kernel and thus is not learned. The second part of our contribution is to propose two refinements. First, we bring to light the fact that one can drastically reduce the complexity of **GBRFF1** by learning a rough approximation of the kernel, yet much simpler and still very effective, using $K$=1. In this scenario, we show that learning the landmarks boils down to finding a single real number in $[-\pi, \pi]$. Then, to speed up the convergence of the algorithm, we suggest to optimize $\boldsymbol{\omega}^t$ after a random initialization from the Fourier transform. We show that a simple gradient descent with respect to this parameter allows a faster convergence with better performance. These two improvements lead to a variant of our original algorithm, called **GBRFF2** and presented in Algorithm 3.3.

**Cheaper landmark learning using the periodicity of the cosine.** As we set $K$=1, the weak learner $h_{\mathbf{a}^t}(\boldsymbol{x})$ is now simply defined as

$$h_{\mathbf{a}^t}(\boldsymbol{x}) = \cos\big(\boldsymbol{\omega}^t \cdot (\boldsymbol{x}^t - \boldsymbol{x})\big)\,,$$

where its parameters are given by $\mathbf{a}^t = (\boldsymbol{\omega}^t, \boldsymbol{x}^t)$. This formulation allows us to eliminate the dependence on the hyper-parameter $K$. Moreover, one can also get rid of $\beta$, because learning the weights $\boldsymbol{q}^t$ (line **7** of Algorithm 3.2) is no more necessary. Instead, since $K$=1, we can see $\alpha^t$ learned at each iteration as a surrogate of these weights. As our weak learner is based on a single random feature, the objective function (line **6**) to learn the landmark at iteration $t$ becomes

$$\boldsymbol{x}^t = \underset{\boldsymbol{x} \in \mathbb{R}^d}{\arg\min}\, f_{\boldsymbol{\omega}^t}(\boldsymbol{x}) = \frac{1}{m} \sum_{i=1}^m \exp\Big( - \tilde{y}_i \cos(\boldsymbol{\omega}^t \cdot (\boldsymbol{x} - \boldsymbol{x}_i))\Big).$$

Let $c \in \{1, \ldots, d\}$ be the index of the $c$-th coordinate of the landmark $\boldsymbol{x}^t$. We can rewrite the objective function as

$$\begin{aligned} f_{\boldsymbol{\omega}^t}(\boldsymbol{x}^t) =& \frac{1}{m} \sum_{i=1}^{m} \exp\Big( -\tilde{y}_i \cos\big(\boldsymbol{\omega}^t \cdot \boldsymbol{x}^t - \boldsymbol{\omega}^t \cdot \boldsymbol{x}_i\big)\Big) \\ =& \frac{1}{m} \sum_{i=1}^{m} \exp\Big( -\tilde{y}_i \cos(\omega_c^t x_c^t + \sum_{j \neq c} \omega_j^t x_j^t - \boldsymbol{\omega}^t \cdot \boldsymbol{x}_i)\Big). \end{aligned}$$

We leverage the periodicity of the cosine function along each direction to find the optimal $c^{\text{th}}$ coordinate of the landmark $x_c^t \in \big[\frac{-\pi}{\omega_c^t}, \frac{\pi}{\omega_c^t}\big]$ that minimizes $f_{\boldsymbol{\omega}^t}(\boldsymbol{x}^t)$ by fixing all the other coordinates. Figure 3.2 illustrates this phenomenon on the two-moons dataset when applying **GBRFF1** with $K$=1. The plots in the first row show the periodicity of the loss represented as repeating diagonal green/yellow stripes (light yellow is associated to the smallest loss). There is an infinite number of landmarks giving such a minimal loss at the middle of the yellow stripes. Thus, by setting one coordinate of the landmark to an arbitrary value, the algorithm is still able at any iteration to find along the second coordinate a value that minimizes the loss (the resulting landmark at the current iteration is depicted by a white cross). The second row shows that such a strategy allows us to get an accuracy of 100% on this toy dataset after 10 iterations. By generalizing this, instead of learning a landmark vector $\boldsymbol{x}^t \in \mathbb{R}^d$, we fix all but one coordinate of the landmark to 0, and then learn a single scalar $b^t \in [-\pi, \pi]$ that minimizes

$$f_{\boldsymbol{\omega}^t}(b^t) = \frac{1}{m} \sum_{i=1}^{m} \exp\big( -\tilde{y}_i \cos(\boldsymbol{\omega}^t \cdot \boldsymbol{x}_i - b^t)\big).$$

**Learning $\boldsymbol{\omega}^t$ for faster convergence.** The second refinement concerns the randomness of the RFF due to vector $\boldsymbol{\omega}^t$. So far, the latter was drawn according to $p$ and then used to learn $b^t$. We suggest instead to fine-tune $\boldsymbol{\omega}^t$ by doing a gradient descent with as initialization the vector drawn from $p$. Supported by the experiments performed in the following, we claim that such a strategy allows us to both speed up the convergence of the algorithm and boost the accuracy. This update requires to add a line of code, just after line **6** of Algorithm 3.2, expressed as a regularized optimization problem:

$$\boldsymbol{\omega}^t = \underset{\boldsymbol{\omega} \in \mathbb{R}^d}{\arg\min} \lambda \|\boldsymbol{\omega}\|_2^2 + \frac{1}{m} \sum_{i=1}^{m} \exp\big( -\tilde{y}_i \cos(\boldsymbol{\omega} \cdot \boldsymbol{x}_i - b^t)\big),$$

its derivative being

$$\frac{\partial f_{\boldsymbol{\omega}}}{\partial \boldsymbol{\omega}}(\boldsymbol{\omega}) = 2\lambda \boldsymbol{\omega} + \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{x}_i \tilde{y}_i \sin(\boldsymbol{\omega} \cdot \boldsymbol{x}_i - b^t) \exp\big( -\tilde{y}_i \cos(\boldsymbol{\omega} \cdot \boldsymbol{x}_i - b^t)\big).$$

## 3.5 Experimental evaluation

The objective of this section is three-fold: first, we aim to bring to light the interest of learning the landmarks rather than fixing them as done in Letarte et al. [2019]; second we study the

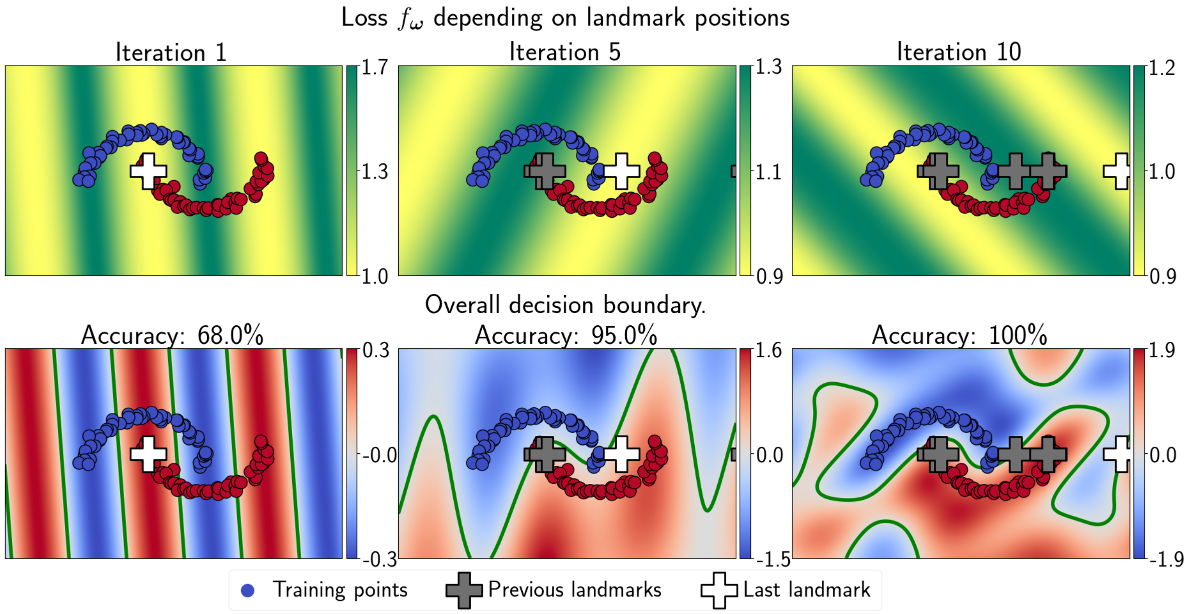


*Figure 3.2:* ***GBRFF1*** *with $K$=1 on the two-moons dataset at different iterations. Top row shows the periodicity of the loss (light yellow indicates the minimal loss). Bottom row shows the resulting decision boundaries between the classes (blue & red) by fixing arbitrarily one coordinate of the landmark and minimizing the loss along the other one.*

impact of the number $K$ of random features; lastly, we perform an extensive experimental comparison of our algorithms. The Python code of all experiments and the data used are publicly available[1].

### 3.5.1 Setting

For **GBRFF1** and **GBRFF2**, we select by cross-validation the hyper-parameter $\gamma \in \frac{2^{\{-2,\ldots,2\}}}{d}$. For **GBRFF2**, we also tune $\lambda \in \{0, 2^{\{-5,\ldots,-2\}}\}$. We compare our two methods with the following algorithms.

- **LGBM** [Ke et al., 2017] is a state-of-the-art gradient boosting method using trees as base predictors. We select by cross-validation the maximum tree depth in $\{1, \ldots, 10\}$ and the L2 regularization parameter $\lambda \in \{0, 2^{\{-5,\ldots,-2\}}\}$.
- **BMKR** [Wu et al., 2017] is a Multiple Kernel Learning method based on gradient boosting with least square loss. It selects at each iteration the best kernel plugged inside an SVR to fit the residuals among 10 RBF kernels with $\gamma \in 2^{\{-4,\ldots,5\}}$ and the linear kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'$. The SVR algorithm is a direct adaptation of the kernelized SVM algorithm described in Section 1.2.2 but where the label to predict is a real number in SVR instead of a class in SVM. We select by cross-validation the SVR parameter $C \in 10^{\{-2,\ldots,2\}}$.
- **GFC** [Oglic and Gärtner, 2016] is a greedy feature construction method based on functional gradient descent. It iteratively refines the representation learned by adding a feature that

[1] The code is available here: `https://leogautheron.github.io`

*Table 3.1: Description of the datasets (n: number of examples, d: number of features, c: number of classes) and the classes chosen as negative (-1) and positive (+1).*

| Name | m | d | c | Label -1 | Label +1 | Name | m | d | c | Label -1 | Label +1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| wine | 178 | 13 | 3 | 2, 3 | 1 | australian | 690 | 14 | 2 | 0 | 1 |
| sonar | 208 | 60 | 2 | M | R | pima | 768 | 8 | 2 | 0 | 1 |
| newthyroid | 215 | 5 | 3 | 1 | 2, 3 | vehicule | 846 | 18 | 4 | van | bus, opel, saab |
| heart | 270 | 13 | 2 | 1 | 2 | german | 1000 | 23 | 2 | 1 | 2 |
| bupa | 345 | 6 | 2 | 2 | 1 | splice | 3175 | 60 | 2 | +1 | -1 |
| iono | 351 | 34 | 2 | g | b | spambase | 4597 | 57 | 2 | 0 | 1 |
| wdbc | 569 | 30 | 2 | B | M | occupancy | 20560 | 5 | 2 | 0 | 1 |
| balance | 625 | 4 | 3 | B, R | L | bankmarketing | 45211 | 51 | 2 | no | yes |

matches the residual function defined for the least squared loss. We use the final representation to learn a linear SVM where $C \in 10^{\{-2,\dots,2\}}$ is selected by cross-validation.

- **PBRFF** [Letarte et al., 2019] described in Section 3.3 that (1) draws with replacement $n_L$ landmarks from the training set; (2) learns a representation of $n_L$ features where each feature is computed using Equation (3.2) based on $K$=10 vectors drawn like our methods from $\mathcal{N}(0, 2\gamma)^d$; (3) learns a linear SVM on the new representation. We select by cross-validation its parameters $\gamma \in \frac{2^{\{-2,\dots,2\}}}{d}$, $\beta \in 10^{\{-2,\dots,2\}}$ and the SVM parameter $C \in 10^{\{-2,\dots,2\}}$.

We consider 16 datasets coming mainly from the UCI repository that we binarized as described in Table 3.1. Note that, we generate for each dataset 20 random 70%/30% train/test splits. Most of them (except occupancy and bankmarketing) are made of a small number of training examples, one of our objective being to show the efficiency of our method in such a scenario. Datasets are pre-processed such that each feature in the training set has 0 mean and unit variance; the factors computed on the training set are then used to scale each feature in the test set. All parameters are tuned by 5-fold cross-validation on the training set by performing a grid search.

Compared to the baseline method **PBRFF**, our two proposed methods **GBRFF1** and **GBRFF2** rely on different strategies in order to obtain an effective and efficient classifier. In the following, we propose different experiments that give some insights on the impact of the strategies used on both the classification accuracy and the computation time.

### 3.5.2 The importance of learning the landmarks in GBRFF1

Our method **GBRFF1** is based on **PBRFF** but is different in two points: *(i)* it learns the model at the same time as the representation instead of first learning the representation and then the model and *(ii)* it learns the landmarks used to build the representation instead of selecting them randomly from the training set. We compare these two methods to a variant called **GBRFF0.5** which is identical to **GBRFF1** except that we do not learn the landmarks in this variant, but we select them randomly as done in **PBRFF**. Figure 3.3 compares these three methods. We see that **GBRFF0.5** is faster than **PBRFF** but that it also leads to a lower accuracy. Thus, simply adapting the two step learning method of **PBRFF** in the one step learning method **GBRFF0.5** degrades the performances while slightly decreasing the computation time. These lower performances might come from the boosting classifier which is less effective than the SVM classifier in this setting. However, when comparing **GBRFF0.5** and **GBRFF1**, we observe that learning the landmarks allows to improve the accuracy which becomes better than the one obtained by **PBRFF**, but at the price of an increase of the computation time which becomes superior than both **GBRFF0.5** and **PBRFF**. These promising results in terms of accuracies motivate us to improve the learning strategy of the landmarks in **GBRFF1** to make it more efficient.

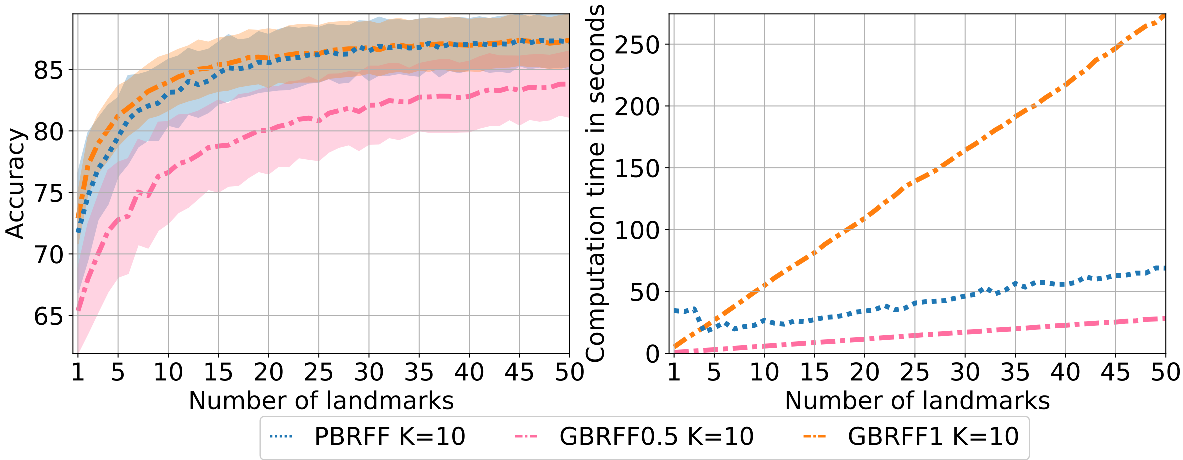


*Figure 3.3: Mean accuracy (left) and sum of computation time using the best parameters found with cross-validation (right) over* 20 *train/test splits and over the* 15 *first datasets (except "bankmarketing") for the three methods* ***PBRFF****,* ***GBRFF0.5*** *and* ***GBRFF1*** *using from* 1 *to* 50 *landmarks.*

### 3.5.3 Improving the efficiency of GBRFF1

A key element of both **PBRFF** and **GBRFF1** is $K$, the amount of random features used for each landmark. We compare the performances and computation time of **GBRFF1** when using different numbers of random features per landmark. The results of this experiment are reported in Figure 3.4. As expected, the accuracy is better using more random features per landmark, but requiring of an increasingly higher computation time. It seems that the higher the amount of random features is, the smaller the gain is in accuracy but the higher the addition to the computation time is. To illustrate this, we present in Figure 3.5 the accuracy divided by the computation time. The results show that **GBRFF1** with $K = 1$ presents the best compromise accuracy/computation time, meaning that even if for a fixed amount of landmarks $T$ we can obtain better performances with a large value of $K$, it is more interesting to set $K = 1$ and use

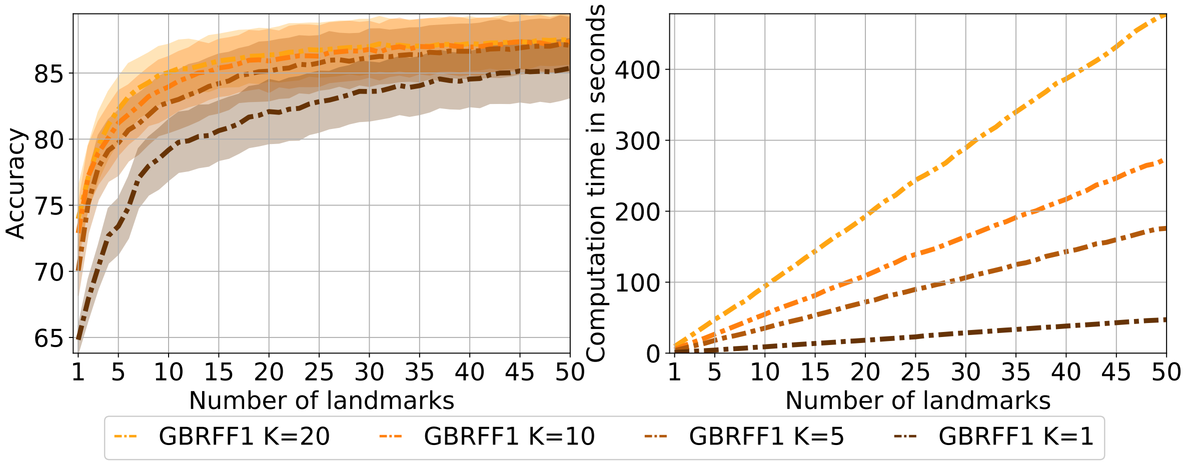


*Figure 3.4: Mean accuracy (left) and sum of computation time using the best parameters found with cross-validation (right) over 20 train/test splits and over the 15 first datasets (except the largest dataset "bankmarketing") for **GBRFF1** with $K \in \{1, 5, 10, 20\}$ random features used per landmark using from 1 to 50 landmarks.*

a large amount of landmarks $T$ to obtain similar performances in less time. This behavior is confirmed by the results presented in Figure 3.6 showing that for different values of $T \times K$, we need to set $K = 1$ and use a large value of $T$ to obtain the best accuracy. To understand why it is better to use a small amount $K$ of random features per landmark, but a large amount of landmarks $T$, we remind the formula of the final predictor of **GBRFF1** for a given example $\boldsymbol{x}$ which is

$$\operatorname{sign}\left(H^0(\boldsymbol{x}) + \sum_{t=1}^{T} \alpha^t \sum_{j=1}^{K} q_j^t \cos\left(\boldsymbol{\omega}_j^t \cdot (\boldsymbol{x}^t - \boldsymbol{x})\right)\right)$$

but that simplifies when $K = 1$ to

$$\operatorname{sign}\left(H^0(\boldsymbol{x}) + \sum_{t=1}^{T} \alpha^t \cos\left(\boldsymbol{\omega}^t \cdot (\boldsymbol{x}^t - \boldsymbol{x})\right)\right).$$

At a given iteration $t$, the objective is to learn the landmark $\boldsymbol{x}^t$, the boosting weight $\alpha^t$ and the random feature weights $\boldsymbol{q}^t$ that fit well the residuals defined by the exponential loss. Consequently, if $K$ increases, so does the amount of constraints imposed at a given iteration to learn the landmark and the boosting weight. A possible explanation is that when $K = 1$, it is simpler to find a landmark and a weight $\alpha^t$ that correctly fit the residuals because both of them are less constrained. On the other hand, when using a large amount of random features, there is no possible solution that fits well the residuals under the constraints imposed by the random features.

The results presented motivate us to build upon **GBRFF1** using the smallest possible amount of random feature $K = 1$ per landmark.

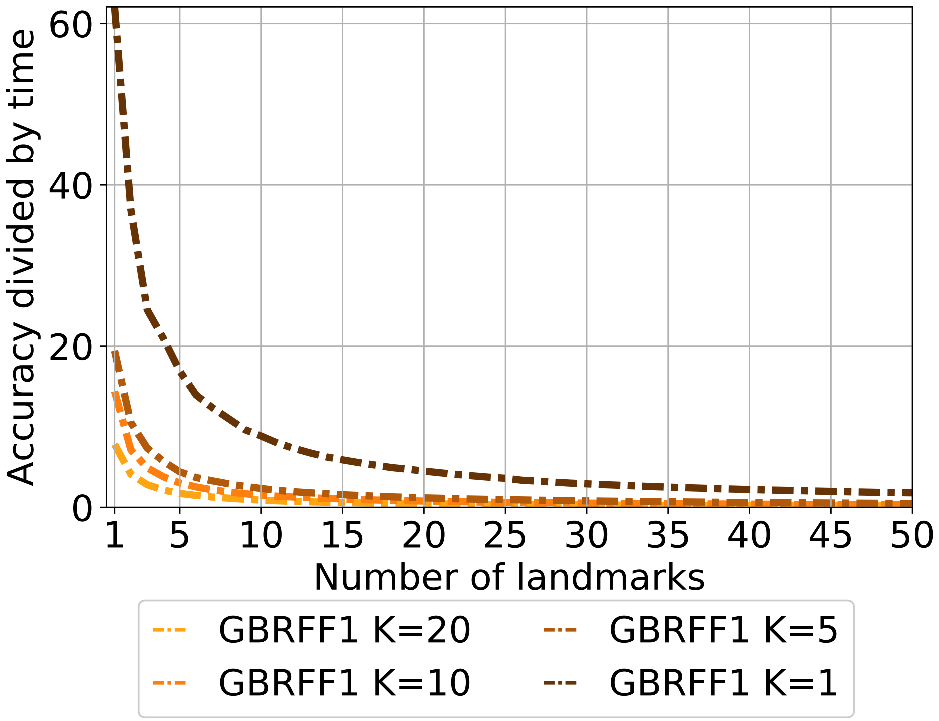


*Figure 3.5: Mean accuracy divided by sum of computation time using the best parameters learned with cross-validation for **GBRFF1** with $K \in \{1, 5, 10, 20\}$ random feature used per landmark using from 1 to 50 landmarks over 20 train/test splits and over the 15 first datasets (except the largest dataset "bankmarketing").*

### 3.5.4 From GBRFF1 to GBRFF2

Our proposed method **GBRFF2** is different from **GBRFF1** as *(i)* it uses a unique random feature per landmark and because *(ii)* the random part of the random feature $\boldsymbol{\omega}$ is learned instead of fixed randomly. We introduce a variant called **GBRFF1.5** identical to **GBRFF2** except for $\boldsymbol{\omega}$ which is not learned but remains fixed randomly. This variant is different from **GBRFF1** because the use of a unique random feature allows to learn a single scalar instead of a landmark vector to obtain the same model as **GBRFF1** with $K = 1$ more efficiently. The comparison in Figure 3.7 between **GBRFF1** with $K = 1$ and **GBRFF1.5** shows as expected that the two methods lead exactly to the same performances but with a much smaller computation time for **GBRFF1.5**. This confirms that when using a unique random feature, it is equivalent to learn a single scalar in $[-\pi, \pi]$ and a landmark vector in $\mathbb{R}^d$, and this is much faster.

On the other hand, **GBRFF2** gives better performances than **GBRFF1.5**, especially with a very small amount of landmarks, but with a larger of the computation time. Compared with **GBRFF1**, **GBRFF2** is faster for $K > 1$ or as fast for $K = 1$, and **GBRFF2** also achieves higher performances, even when using $K = 20$ random features for **GBRFF1**.

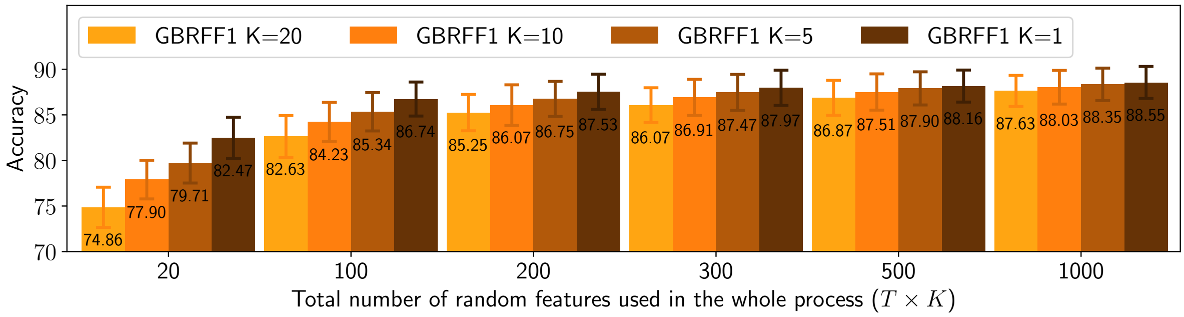


*Figure 3.6: Mean results over the 16 datasets w.r.t. the same total amount of random features $T \times K$ for $K \in \{1, 5, 10, 20\}$, with $T$ the amount of boosting iterations.*

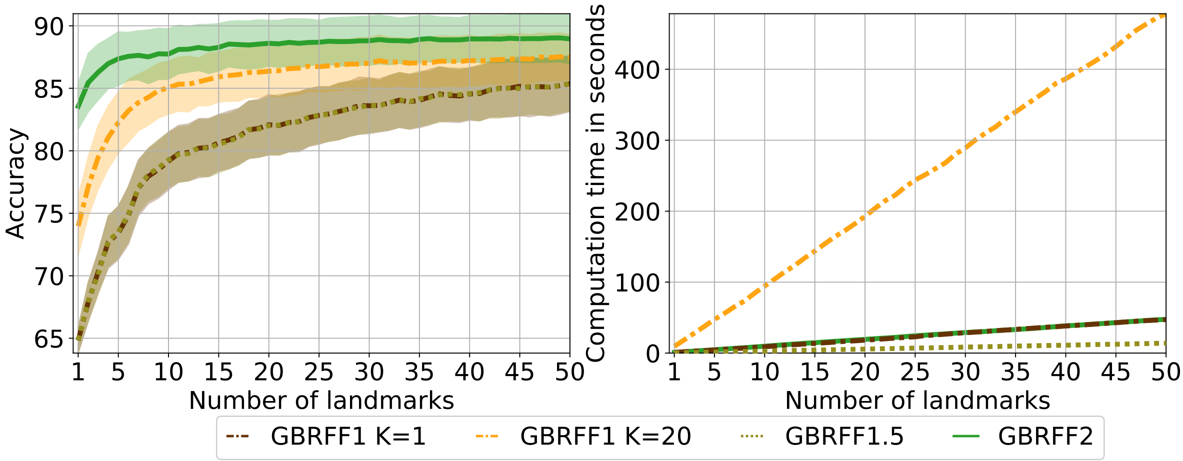


*Figure 3.7: Mean accuracy (left) and sum of computation time using the best parameters found with cross-validation (right) over 20 train/test splits and over the 15 first datasets (except the largest dataset "bankmarketing") for* ***GBRFF1****,* ***GBRFF1.5*** *and* ***GBRFF2*** *using from 1 to 50 landmarks.*

### 3.5.5 Influence of learning the landmarks

We present in Figure 3.8 the behavior of the three methods that make use of landmarks and RFF, that is **PBRFF**, **GBRFF1** and **GBRFF2**. With more than 25 landmarks, **PBRFF** and **GBRFF1** show similar mean accuracy and reach about 87.5% after 50 iterations. However, for a small set of landmarks (in particular smaller than 25) **GBRFF1** is consistently superior by about 1 point higher than **PBRFF**, showing the interest of learning the landmarks. But the certainly most striking result comes from the performance of our variant **GBRFF2** which outperforms the two competing methods. This is particularly true for a small amount of landmarks. Notice that **GBRFF2** is able to reach its maximum with about 20 landmarks, while **GBRFF1** and **PBRFF** require more iterations without reaching the same performance. This definitely shows the benefit of learning the random features compared to drawing them randomly.

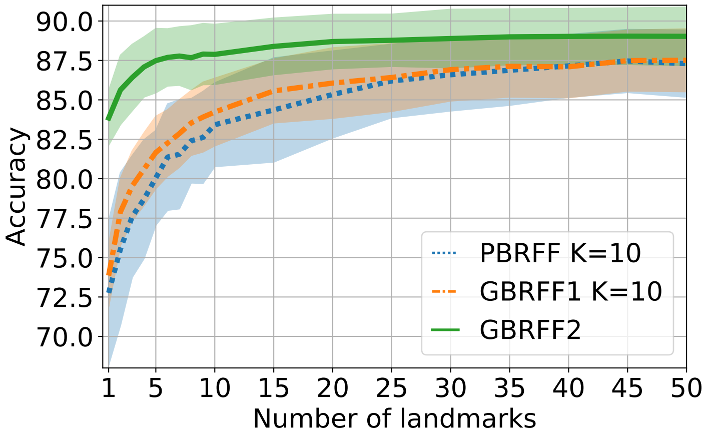


*Figure 3.8: Mean test accuracy over* 20 *train/test splits over the* 16 *datasets. We train the three methods using from* 1 *to* 50 *landmarks.*

### 3.5.6 Influence of the number of examples on the computation time

The specificities of **GBRFF2** come from the number of random features $K$ set to 1 at each iteration and the learning of $\boldsymbol{\omega}^t$. We have already shown in Figure 3.2 that this allows us to get better results. We study in this section how **GBRFF2** scales compared to the other methods. To do so, we consider artificial datasets with an increasing number of examples (generated with scikit-learn [Pedregosa et al., 2011] library's `make_classification` function). The initial size is set to 150 examples, and we successively generate datasets with a size equal to the previous dataset size multiplied by 1.5. Here, we do not split the datasets in train and test as we are not interested in the accuracy. We report the time in seconds necessary to train the models and to predict the labels on the whole datasets. The parameters are fixed as follows: $C = 1$ for the methods using SVM or SVR; the tree depth is set to 5 for **LGBM**; $K = 10$, $\gamma = \frac{1}{d}$, and $\beta = 1$ for **PBRFF** and **GBRFF1**; $\gamma = \frac{1}{d}$ and $\lambda = 0$ for **GBRFF2**. All the methods are run with 100 iterations (or landmarks) and are not run on datasets requiring more than 1000 seconds of execution time (because larger datasets requiring more than 1000 seconds by the fastest method do not fit in the RAM memory of the computer used for the experiments). We report the results in Figure 3.9.

We first recall that **GBRFF2** learns at each iteration a random feature and a landmark while **GBRFF1** only learns the landmark and **PBRFF** draws them randomly. Thus, **GBRFF1** should present higher computation times compared to **PBRFF**. However, for datasets with a number of examples larger than $20,000$, **GBRFF1** becomes cheaper than **PBRFF**. This is due to the fact that the SVM classifier learned by **PBRFF** does not scale as well as gradient boosting-based methods. The two-step method **GFC** is in addition also slower than **GBRFF1**. This shows the computational advantage of having a one-step procedure to learn both the representation and the final classifier. When looking at the time limit of 1000 seconds, both **GBRFF1** and **GBRFF2** are the fastest kernel-based methods compared to **BMKR**, **GFC** and **PBRFF**. This shows the efficiency of learning kernels in a greedy fashion. We also see that

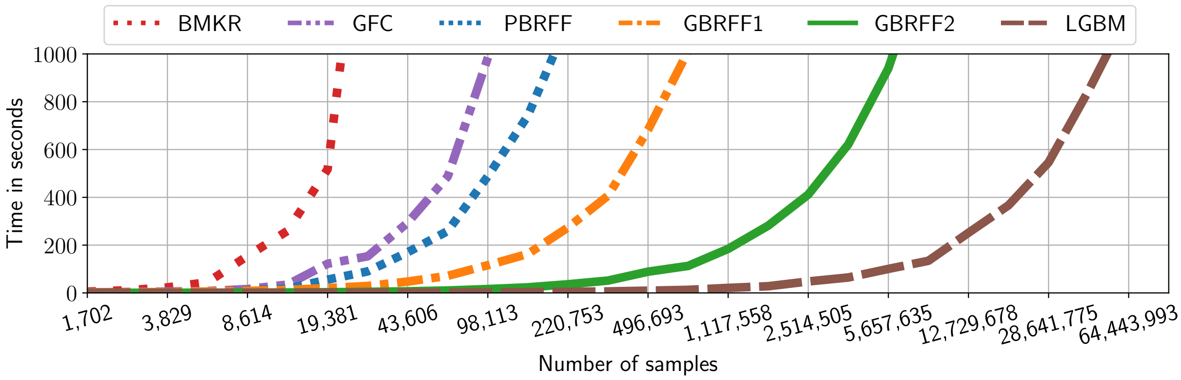


*Figure 3.9: Computation time in seconds required to train and test the six methods with fixed parameters on an artificial dataset having an increasing number of examples. The whole dataset is used for training and testing, and a method requiring more than* 1000 *seconds at a given step is not trained on the larger datasets.*

**GBRFF2** performs faster than **GBRFF1** for any number of examples. At the limit of 1000 seconds, it is able to deal with datasets that are 10 times larger than **GBRFF1**, due to the lower complexity of the learned weak learner used in **GBRFF2**. Finally, **GBRFF2** is globally the second-fastest method behind the gradient boosting method **LGBM** that uses trees as base classifiers.

### 3.5.7 Performance comparison between all methods

Table 3.2 presents for each dataset the mean results over the 20 splits using 100 iterations/landmarks for each method. Due to the size of the dataset "bankmarketing", we do not report the results of the algorithms that do not converge in time for this dataset, and we compute the average ranks and mean results over the other 15 datasets. In terms of accuracy, **GBRFF2** shows very good results compared with the state-of-the-art as it obtains the best average rank among the six methods and on average the best mean accuracy leaving apart "bankmarketing". Interestingly, our method is the only kernel-based one that scales well enough to be applied to this latter dataset.

### 3.5.8 GBRFF2 is able to learn complex decision boundaries that generalizes well on small datasets

In this last experiment, we focus on **LGBM** and **GBRFF2** which have been shown to be the two best performing methods in terms of accuracy and execution time. Even if **BMKR** is among the three best methods in terms of accuracy, we do not consider it for this experiment due to its poor execution time. Learning a classifier based on non-linear kernels through **GBRFF2** has the advantage of being able to capture non-linear decision surfaces, whereas **LGBM** is not well suited for this because it uses trees as base learner. To illustrate this advantage, we consider three synthetics 2D datasets with non-linearly separable classes. The first one, called

*Table 3.2: Mean test accuracy ± standard deviation over 20 random train/test splits. A '-' in the last row indicates that the algorithm did not converge in time on this dataset. Average ranks and mean results are computed over the first 15 datasets.*

| Dataset | BMKR | GFC | PBRFF | GBRFF1 | LGBM | GBRFF2 |
|---|---|---|---|---|---|---|
| wine | **99.5** ± 1.0 | 99.3 ± 1.1 | 98.1 ± 2.1 | 98.3 ± 1.5 | 96.6 ± 3.2 | 98.5 ± 1.6 |
| sonar | 78.8 ± 7.2 | 76.6 ± 3.2 | 76.7 ± 5.2 | 81.8 ± 3.5 | 82.4 ± 4.3 | **83.0** ± 5.0 |
| newthyroid | 96.5 ± 1.7 | 96.5 ± 2.1 | 96.5 ± 1.5 | 95.3 ± 2.2 | 94.8 ± 2.9 | **96.9** ± 2.1 |
| heart | **85.6** ± 4.0 | 79.4 ± 4.5 | 85.4 ± 3.5 | 83.6 ± 4.0 | 83.0 ± 3.5 | 83.1 ± 4.0 |
| bupa | 68.1 ± 4.9 | 64.7 ± 3.2 | 69.0 ± 4.2 | 70.3 ± 4.9 | **72.0** ± 3.3 | 71.2 ± 4.5 |
| iono | **94.2** ± 1.4 | 91.5 ± 2.3 | 94.2 ± 1.8 | 88.2 ± 2.3 | 93.3 ± 2.5 | 89.2 ± 2.1 |
| wdbc | 96.1 ± 1.2 | 95.8 ± 1.3 | 96.5 ± 1.1 | 96.8 ± 1.1 | 95.8 ± 1.5 | **97.3** ± 1.2 |
| balance | 96.0 ± 1.2 | 95.1 ± 2.0 | **98.9** ± 1.1 | 97.7 ± 0.7 | 93.5 ± 2.6 | 97.7 ± 0.6 |
| australian | 85.9 ± 2.0 | 80.9 ± 2.4 | 84.6 ± 2.3 | 86.7 ± 1.7 | 85.5 ± 1.9 | **86.9** ± 1.9 |
| pima | 76.4 ± 2.0 | 68.7 ± 2.6 | 76.1 ± 2.5 | 76.5 ± 2.7 | 75.5 ± 2.7 | **77.1** ± 2.5 |
| vehicle | 96.6 ± 1.3 | 95.9 ± 0.8 | 96.5 ± 1.4 | 96.3 ± 1.2 | 96.7 ± 1.0 | **97.1** ± 1.0 |
| german | 72.3 ± 1.8 | 64.3 ± 2.8 | 72.4 ± 1.4 | 73.7 ± 1.6 | 73.5 ± 1.7 | **74.0** ± 1.3 |
| splice | 87.5 ± 1.0 | 87.0 ± 1.0 | 83.5 ± 0.7 | 83.9 ± 1.1 | **97.0** ± 0.5 | 92.4 ± 0.8 |
| spambase | 93.5 ± 0.4 | 91.3 ± 0.6 | 91.6 ± 0.7 | 90.7 ± 0.7 | **95.6** ± 0.4 | 92.8 ± 0.6 |
| occupancy | **99.3** ± 0.1 | 98.9 ± 0.7 | 98.9 ± 0.1 | 98.8 ± 0.1 | 99.3 ± 0.1 | 98.9 ± 0.1 |
| Mean | 88.4 ± 2.1 | 85.7 ± 2.0 | 87.9 ± 2.0 | 87.9 ± 2.0 | 89.0 ± 2.1 | **89.1** ± 2.0 |
| Average Rank | 2.88 | 4.94 | 3.75 | 3.81 | 3.44 | **2.19** |
| bankmarketing | - | - | - | 89.7 ± 0.2 | **90.8** ± 0.2 | 90.0 ± 0.2 |

"swiss", represents two spirals of two classes side by side. The second one, namely "circles", consists of four circles with the same center and an increasing radius by alternating the class of each circle. The third dataset, called "board", consists of a four by four checkerboard with alternating classes in each cell. Here, both **LGBM** and **GBRFF2** are run for 1000 iterations to ensure their convergence and parameters are tuned by cross-validation as previously.

Figure 3.10 gives evidence that **GBRFF2** is able to achieve much better results than **LGBM** when using only a small amount of training examples, *i.e.*, 500 or less. Furthermore, if we look at the decision boundaries and their associated performances at train and test time, we can see that **LGBM** is prone to overfit the training data compared to our approach, showing a drastic drop in performance between learning and testing. The learned decision boundaries are also smoother with **GBRRF2** than with **LGBM**. These experiments show the advantage of having a non-linear weak learner in a gradient boosting approach.

## 3.6 Conclusion and perspectives

In this chapter, we take advantages of two machine learning approaches, gradient boosting and random Fourier features, to derive a novel algorithm that jointly learns a compact representation

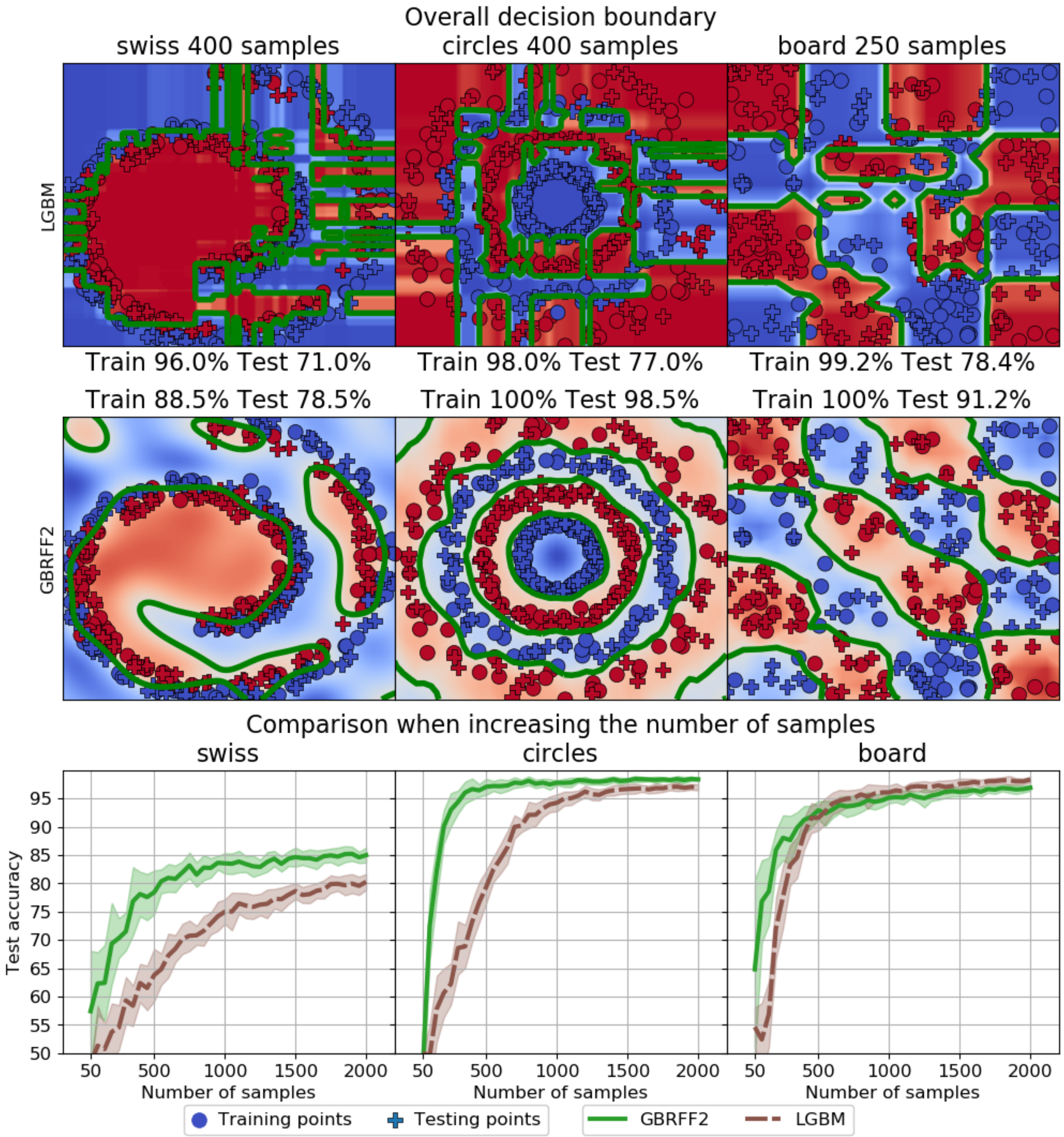


*Figure 3.10: Comparison of **LGBM** and **GBRFF2** on three synthetic datasets in terms of classification accuracy and decision boundaries (upper part of the figure) and in terms of performance w.r.t. the number of examples (last row of plots).*

and a model based on random features. Building on the recent work by Letarte et al. [2019], we learn a kernel by approximating it as a weighted sum of RFF [Rahimi and Recht, 2008]. The originality is that we learn such kernels so that the representation and the classifier are jointly optimized. We show that we can benefit from a performance boost in terms of accuracy and computation time by considering each weak learner as a single trigonometric feature and learning the random part of the RFF. The experimental study shows the competitiveness of our method with state-of-the-art boosting and kernel learning methods. In particular, our method is able to learn non-linear decision boundaries which generalize well in the presence of few labeled examples.

The optimization of the random feature and of the landmark at each iteration can be computationally expensive when the number of iterations is large. A promising future line of research to speed-up the learning is to derive other kernel approximations where these two parameters can be computed with a closed-form solution. Other perspectives regarding the scalability include the use of standard gradient boosting tricks [Ke et al., 2017] such as sampling or learning the kernels in parallel.

Chapter 4

# Representations Learning for Unsupervised Domain Adaptation

**This chapter is based on the following publication**

Léo Gautheron, Ievgen Redko and Carole Lartizien. Feature Selection for Unsupervised Domain Adaptation using Optimal Transport. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2018, Dublin, Ireland [Gautheron et al., 2018b].

**Abstract**

In this chapter, we address the difficult problem of unsupervised domain adaptation where the learner does not benefit from any label of the target domain. We build upon a recent theoretical analysis of optimal transport in domain adaptation and show that it can directly suggest a feature selection procedure leveraging the shift between the source and target domains. We propose a novel algorithm that aims to sort features by their similarity across the domains, where the order is obtained by analyzing the coupling matrix representing the solution of the proposed optimal transportation problem. We evaluate our method on a well-known benchmark dataset and illustrate its capability of selecting correlated features leading to better classification performances. Furthermore, we show that the proposed algorithm can be used as a pre-processing step for existing domain adaptation techniques ensuring an important speed-up in terms of the computational time while maintaining comparable results. Finally, we validate our algorithm on clinical imaging databases for computer-aided diagnosis task with promising results.

## 4.1 Introduction

The majority of well-known machine learning algorithms used in real-world applications are built upon the common strategy often known as empirical risk minimization, as described in Chapter 1. This strategy suggests that a classifier that minimizes the loss over the observed dataset is expected to generalize and thus to perform well on any other example coming from the same probability distribution. However, this assumption is often violated in practice where a dataset may be different from new unseen data collected afterwards. For instance, one may consider a computer aided diagnostic system developed to detect a specific disease in patients. It is quite intuitive to suggest that the data collected on a set of patients at a certain hospital to build the system will present differences with the data collected at another hospital due to the differences in the acquisition process and on the different population of patients. Even if the system may produce good detection results in the first hospital where it was developed, it might present poor detection performance for the other hospital because of the differences in the data between the two hospitals. In this Chapter, we are interested in this problem where the training set has enough data to build a model, but where the test set as only a few or even no labels. In this setting, we must rely on the data from the training set to build a model deployed on the test set, which may fail on the test because the data are acquired in different conditions between the two domains. In order to tackle this problem, a learning paradigm called domain adaptation was proposed by Ben-David et al. [2007].

The main goal of domain adaptation is to provide methodological frameworks and algorithms that allow to reuse a classifier learned in one area, usually called *source domain*, in a different yet similar area usually called *target domain*. According to the domain adaptation theory presented in [Ben-David et al., 2007, 2010] (see also Redko et al. [2019] for a survey), the efficiency of a given adaptation algorithm depends on its capacity to reduce the discrepancy between the probability distributions of the considered source and target datasets and on the existence of a good hypothesis (or classifier) that can minimize both source and target error functions. While finding this optimal hypothesis is a very difficult problem, most domain adaptation algorithms concentrate solely on reducing the discrepancy between two domains based on the observed examples. To this end, several papers [Uguroglu and Carbonell, 2011, Persello and Bruzzone, 2015, Li et al., 2016, Yin et al., 2017] proposed to solve the domain adaptation problem by addressing it as a feature selection task. Indeed, for the general adaptation scenario, it is reasonable to assume that the shift between the source and target domains may be caused by a changing behavior of a subset of features that characterize the data in both domains. In this case, identifying these features can help to reduce the discrepancy between the source and target domains and to allow efficient adaptation.

In this chapter, we propose to learn a joint representation between the source and target through the lens of a feature selection algorithm. Dedicated to deal with the for unsupervised domain adaptation setting, our algorithm allows to rank features based on their similarity across the source and target domains. Our key underlying idea is to solve the optimal transportation problem (recalled in Section 1.3.3) between the marginal distributions of features in the two domains in order to obtain a coupling matrix given by their joint probability distribution. The goal, then, is to use this coupling matrix to identify the most correlated features by analyzing the diagonal of the coupling matrix where higher coupling values indicate strong correlations between the source and target features. Contrary to the state-of-the-art methods that proceed by learning a new richer feature representation before identifying the invariant features, our method performs feature selection directly in the input space. This choice leads to more interpretable results and to a better understanding of the adaptation phenomenon as transformed features cannot directly point out to those descriptors that vary between the two domains. Furthermore, the shifted features identified by our method can be eliminated in order to speed-up domain adaptation algorithms whose running time often inherently depends on the dimensionality of the input data. This latter point is quite important as domain adaptation algorithms are often deployed for high-dimensional data arising from computer vision applications.

Despite its advantages, our method does not aim to outperform the state-of-the-art classification results obtained by powerful feature transformation domain adaptation methods as most of them use a very rich class of mappings to find a new data representation. To this end, the foremost goal of this chapter is to show that the proposed feature selection method is not a competitor of the state-of-the-art algorithms but is a complementary tool that provides important benefits both in terms of computational efficiency and better understanding of data. All the results presented in our chapter are given in order to illustrate this rather than its superiority in terms of classification accuracy.

**Organization of the chapter.** In Section 4.2 we present a short state-of-the-art on feature selection methods in domain adaptation. Section 4.3 is devoted to the introduction of basic elements related to the optimal transportation theory that are used later. In Section 4.4, we show how a theoretical analysis of domain adaptation with optimal transport can be used to derive a new adaptation algorithm based on feature selection. Based on this, we describe the proposed method and the details of its algorithmic implementation. Section 4.5 presents experimental evaluations of the proposed method on both a benchmark computer vision dataset and a clinical imaging database for computer-aided diagnosis task. Section 4.6 summarizes our chapter by outlining its main contributions and giving the possible future perspectives of this work.

## 4.2 Related work

As classical feature selection methods [Guyon and Elisseeff, 2003] are not designed to work well under the assumption of distribution's shift, several methods were specifically proposed in the literature for feature selection in the context of domain adaptation. For instance, in [Li et al., 2016], the authors search a latent low-dimensional subspace for two domains by jointly preserving the data structure and by selecting a subset of the latent features through a row-sparsity inducing regularization. While being quite effective in terms of classification results, this method, however, has two important drawbacks. First, it does not identify the original features that contribute to efficient adaptation but rather learns their embedding where the distributions' discrepancy is minimized. Second, its optimization procedure makes use of eigenvalue decomposition which has a high computational cost in large-scale applications. In [Yin et al., 2017], the authors learn a least squares SVM in order to further remove the features that incur the smallest loss of the classification margin between the classes. Another paper from Persello and Bruzzone [2015] proposes to solve an optimization problem with two terms: the first one maximizes the relevance between source features and labels using the Hilbert-Schmidt Independence Criterion while the second term minimizes the shift between the domains using kernel embeddings. Contrary to our algorithm, the above mentioned methods are supervised as they both use annotations in the target domain. Finally, the method that is the most similar to ours is the feature selection algorithm for transfer learning presented by Uguroglu and Carbonell [2011]. In this latter paper, the authors use a parametric maximum mean discrepancy distance in order to find a weight matrix that allows to identify invariant and shifting features in the original space. As we will show in Section 4.4.1, this method and our contribution are closely related from a theoretical point of view, even though our method remains much more computationally attractive.

## 4.3 Preliminary knowledge

Optimal transport has been already described in Section 1.3.3, and mainly relies on the two Equations (1.4) and (1.6) recalled bellow.

$$\boldsymbol{\gamma}^* = \underset{\boldsymbol{\gamma} \in \Pi(\widehat{\boldsymbol{\mathcal{D}}}^S_{\mathcal{X}}, \widehat{\boldsymbol{\mathcal{D}}}^T_{\mathcal{X}})}{\arg\min} \langle \boldsymbol{\gamma}, \boldsymbol{C} \rangle_F, \tag{4.1}$$

$$\boldsymbol{\gamma}^* = \underset{\boldsymbol{\gamma} \in \Pi(\widehat{\boldsymbol{\mathcal{D}}}^S_{\mathcal{X}}, \widehat{\boldsymbol{\mathcal{D}}}^T_{\mathcal{X}})}{\arg\min} \langle \boldsymbol{\gamma}, \boldsymbol{C} \rangle_F - \frac{1}{\lambda} E(\boldsymbol{\gamma}), \tag{4.2}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius dot product, $\Pi(\widehat{\boldsymbol{\mathcal{D}}}^S_{\mathcal{X}}, \widehat{\boldsymbol{\mathcal{D}}}^T_{\mathcal{X}}) = \{\boldsymbol{\gamma} \in \mathbb{R}_+^{m \times n} | \boldsymbol{\gamma} \mathbf{1} = \widehat{\boldsymbol{\mathcal{D}}}^S_{\mathcal{X}}, \boldsymbol{\gamma}^\top \mathbf{1} = \widehat{\boldsymbol{\mathcal{D}}}^T_{\mathcal{X}}\}$ is a set of doubly stochastic matrices and $\boldsymbol{C}$ is the cost matrix where $C_{ij}$ is the cost between $\boldsymbol{x}_i^S \in \boldsymbol{X}^S$ and $\boldsymbol{x}_j^T \in \boldsymbol{X}^T$ which defines the energy needed to move a probability mass from $\boldsymbol{x}_i^S$ to $\boldsymbol{x}_j^T$. We abbreviate the problem given in Equations (4.1) and (4.2) respectively as **OT** and **OT2**. The use of optimal transport for domain adaptation has been studied for the first time by Courty et al. [2014]. In this work, the authors present a new variant of optimal transport (abbreviated **OT3**) based on Equation (4.2) by adding a class regularization $\ell_{\frac{1}{2},1}$:

$$\boldsymbol{\gamma}^* = \underset{\boldsymbol{\gamma} \in \Pi(\widehat{\boldsymbol{\mathcal{D}}}^S_{\mathcal{X}}, \widehat{\boldsymbol{\mathcal{D}}}^T_{\mathcal{X}})}{\arg\min} \langle \boldsymbol{\gamma}, \boldsymbol{C} \rangle_F - \frac{1}{\lambda} E(\boldsymbol{\gamma}) + \eta \Omega(\boldsymbol{\gamma}), \tag{4.3}$$

where the $\Omega(\boldsymbol{\gamma}) = \sum_j \sum_l \|\boldsymbol{\gamma}_{I_l j}\|_1^{1/2}$ term prevents the source instances with different labels from being transported to the same target instance. $\boldsymbol{\gamma}_{I_l j}$ is a vector which is a subset of the $j$th column of $\boldsymbol{\gamma}$ and where the indexes of the selected rows are given in $I_l$ which lists the indexes of the examples in $\boldsymbol{X}^S$ with a label equal to $l$ in $\boldsymbol{Y}^S$, and $j$ goes through the indexes of the examples in $\boldsymbol{X}_T$.

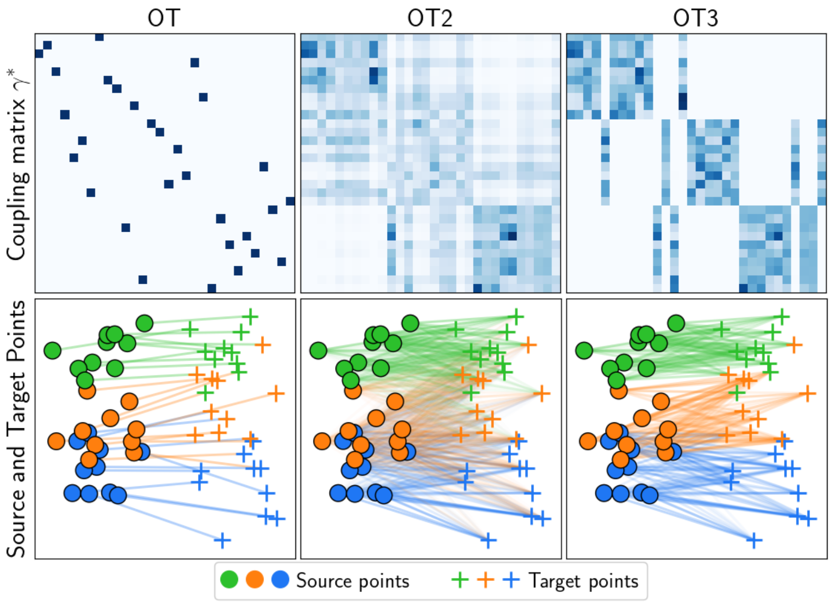


*Figure 4.1: Comparison of the 3 variants of optimal transport:* ***OT*** *on the left,* ***OT2*** *in the middle ($\lambda = 1$),* ***OT3*** *on the right ($\lambda = 1$, $\eta = 1$). First row shows $\boldsymbol{\gamma}^*$ with the higher coupling values seen as darkest blue. The second row shows the source and target points composed of 3 classes in 3 colors. The coupling between them are shown as segments.*

Using the optimal coupling matrix $\boldsymbol{\gamma}^*$ found with Equations (4.1), (4.2) or (4.3), the authors propose to transport the source examples by solving for each of them:

$$\boldsymbol{x}_i^{Sa} = \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathbb{R}^d} \sum_{j=1}^{n} \gamma_{ij}^* c(\boldsymbol{x}, \boldsymbol{x}_j^T). \tag{4.4}$$

In the case of the squared Euclidean distance, the closed form solution of this problem can be written as [Courty et al., 2014]:

$$\boldsymbol{X}^{Sa} = \operatorname{diag}\left((\boldsymbol{\gamma}^* \mathbf{1})^{-1}\right) \boldsymbol{\gamma}^* \boldsymbol{X}^T. \tag{4.5}$$

When $\widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^S$ and $\widehat{\boldsymbol{\mathcal{D}}}_{\mathcal{X}}^T$ are uniform (in practice, this is always the case for us), Equation (4.5) is simplified to

$$\boldsymbol{X}^{Sa} = m\boldsymbol{\gamma}^* \boldsymbol{X}^T. \tag{4.6}$$

With this expression, each source instance is represented as the weighted barycenter of the target instances with which it has the highest values in $\boldsymbol{\gamma}^*$.

We give a graphical comparison of the **OT**, **OT2** and **OT3** algorithms in Figure 4.1. We see that the basic **OT** associates one target instance to one source instance while with **OT2** each source point's mass is divided and transported to its closest target points. By adding the class regularization **OT3**, we prevent the algorithm from transporting the mass of source instances of different classes to the same target instance.

## 4.4 Proposed approach

In this section, we present our main contribution. We start by formally introducing a theoretical result that we use to derive our algorithm.

### 4.4.1 Theoretical insight

The use of optimal transport in domain adaptation was first theoretically analyzed by Redko et al. [2017]. In this paper, the authors proved that under some mild assumptions imposed on the form of the transport cost function, and given any convex loss $h \in \mathcal{H}$ with $h : \mathcal{X} \to [0, 1]$, the true target risk $R^T(h)$ and the true source risk $R^S(h)$ can be related through the following inequality

$$R^T(h) \leq R^S(h) + W(\mathcal{D}^S, \mathcal{D}^T) + \lambda, \tag{4.7}$$

where $\lambda$ is the combined error of the ideal hypothesis $h^*$ that minimizes $R^S(h) + R^T(h)$ and $W$ is the Wasserstein distance computed using the optimal transport plan as defined in Equation (1.5). This result shows that in order to upper-bound the error of a classifier in the target domain, one has to minimize the source error function and the discrepancy between the source and target distributions given by the Wasserstein distance.

Below, we use this result as a starting point in order to develop our approach. To this end, we first notice that the source and target domains can be equivalently seen as 2-dimensional product spaces $\mathcal{X}^S \times \mathcal{F}^S$ and $\mathcal{X}^T \times \mathcal{F}^T$, where $\mathcal{X}^S$ (resp. $\mathcal{X}^T$) and $\mathcal{F}^S$ (resp. $\mathcal{F}^T$) denote the source (resp. target) instance and feature spaces. In this case, the probability distributions $\mathcal{D}^S$ and $\mathcal{D}^T$ are also product measures supported on $\mathcal{X}^S \times \mathcal{F}^S$ and $\mathcal{X}^T \times \mathcal{F}^T$ and can be written as $\mathcal{D}^S_\mathcal{X} \times \mathcal{D}^S_\mathcal{F}$ and $\mathcal{D}^T_\mathcal{X} \times \mathcal{D}^T_\mathcal{F}$, respectively. Using the results proved by [Talagrand, 1995] for concentration of measures in product spaces, we can upper bound the Wasserstein distance between $\mathcal{D}^S$ and $\mathcal{D}^T$ as follows:

$$W(\mathcal{D}^S, \mathcal{D}^T) \leq W(\mathcal{D}^S_\mathcal{F}, \mathcal{D}^T_\mathcal{F}) + \int_{\mathcal{F}_S} W(\mathcal{D}^S_\mathcal{X} | \mathcal{D}^S_\mathcal{F}, \mathcal{D}^T_\mathcal{X}) d\mathcal{D}^S_\mathcal{F}.$$

In this inequality, the measures $\mathcal{D}^S_\mathcal{F}$ (resp. $\mathcal{D}^T_\mathcal{F}$) and $\mathcal{D}^S_\mathcal{X}$ (resp. $\mathcal{D}^T_\mathcal{X}$) can be used interchangeably. Now, by plugging it into the learning bound of Equation (4.7), we obtain

$$R^T(h) \leq R^S(h) + W(\mathcal{D}^S_\mathcal{F}, \mathcal{D}^T_\mathcal{F}) + \int_{\mathcal{F}_S} W(\mathcal{D}^S_\mathcal{X} | \mathcal{D}^S_\mathcal{F}, \mathcal{D}^T_\mathcal{X}) d\mathcal{D}^S_\mathcal{F} + \lambda.$$

This inequality shows that when one considers probability measures over a product space of instances and features spaces, a successful adaptation necessitates the minimization of the discrepancy between the feature distributions $\mathcal{D}^S_\mathcal{F}$, $\mathcal{D}^T_\mathcal{F}$ as well as that of the instance distributions $\mathcal{D}^S_\mathcal{X}$, $\mathcal{D}^T_\mathcal{X}$ conditionally on the source features measure $\mathcal{D}^S_\mathcal{F}$. Thus, it naturally leads to a two-stage procedure where the first goal is to reduce the discrepancy between the feature sets of the two domains while the second is to apply an appropriate domain adaptation algorithm between their instances described by an optimal set of features obtained at the first stage.

In what follows, we introduce our method based on the idea of finding a coupling that aligns the distributions of features across the source and target domains. As suggested by the obtained bound, the selected features minimizing the $W(\mathcal{D}^S_\mathcal{F}, \mathcal{D}^T_\mathcal{F})$ can be used then by a domain adaptation algorithm applied to the source and target examples of a reduced dimensionality. The Wasserstein distance here can be replaced, in practice, by the popular maximum mean discrepancy distance [Gretton et al., 2012] often used in domain adaptation as both of them belong to a larger class of integral probability metrics defined over different functional classes. In this case, the feature selection algorithm proposed by Uguroglu and Carbonell [2011][1] also indirectly minimizes the discrepancy between the marginals $\mathcal{D}^S_\mathcal{F}$ and $\mathcal{D}^T_\mathcal{F}$. Nevertheless, the computational complexity of the proposed optimization procedure is polynomial thus making its use prohibitive in real-world applications.

### 4.4.2 Problem setup

Until now the optimal transport was used in order to align the empirical measures $\widehat{\boldsymbol{\mathcal{D}}}^S_\mathcal{X}$ and $\widehat{\boldsymbol{\mathcal{D}}}^T_\mathcal{X}$ defined based on the observable datasets $\boldsymbol{X}^S \in \mathbb{R}^{m\times d}$ and $\boldsymbol{X}^T \in \mathbb{R}^{n\times d}$. The interpolation step performed using Equation (4.6) aims at re-weighting the source instances so that their distribution matches the one of the target examples. The geometric interpretation is that, to minimize the divergence between $\mathcal{D}^S$ and $\mathcal{D}^T$, we can associate the source examples with the target examples based on the highest coupling values.

As mentioned in the previous section, the idea of our method is to go from the example space to the feature space. To this end, we now consider that $\boldsymbol{X}^S$ and $\boldsymbol{X}^T$ are drawn from 2-dimensional product spaces $\mathcal{X}^S \times \mathcal{F}^S$ and $\mathcal{X}^T \times \mathcal{F}^T$, where $\mathcal{X}^S, \mathcal{X}^T \subseteq \mathbb{R}^d$ while $\mathcal{F}^S \subseteq \mathbb{R}^m$ and $\mathcal{F}^T \subseteq \mathbb{R}^n$. In this case, we can define two empirical probability measures

$$\widehat{\boldsymbol{\mathcal{D}}}^S_\mathcal{F} = \frac{1}{d}\sum_{i=1}^{d} \boldsymbol{\delta}_{f_i^S}, \qquad \text{and} \qquad \widehat{\boldsymbol{\mathcal{D}}}^T_\mathcal{F} = \frac{1}{d}\sum_{i=1}^{d} \boldsymbol{\delta}_{f_i^T},$$

based on the source and target features $\{f_i^S\}_{i=1}^d \in \mathcal{F}^S$, $\{f_i^T\}_{i=1}^d \in \mathcal{F}^T$, respectively. Our goal now is to transport $\widehat{\boldsymbol{\mathcal{D}}}^S_\mathcal{F}$ to $\widehat{\boldsymbol{\mathcal{D}}}^T_\mathcal{F}$ by solving the entropy regularized optimal transportation problem given as follows:

$$\gamma^{*f} = \underset{\gamma^f \in \Pi(\widehat{\boldsymbol{\mathcal{D}}}^S_\mathcal{F}, \widehat{\boldsymbol{\mathcal{D}}}^T_\mathcal{F})}{\arg\min} \langle \gamma^f, \boldsymbol{C}^f \rangle_F - \frac{1}{\lambda} E\left(\gamma^f\right), \tag{4.8}$$

where $C^f_{ij} = \|f_i^S - f_j^T\|_2^2$.

In what follows, we show that the solution of this problem can lead to a principally different domain adaptation method that is based on a feature selection approach rather than on the original instance re-weighting one.

[1] Unfortunately, we were unable to use this method as a baseline in our experiments due to the lack of implementation details in their paper and the absence of a publicly available code.

### 4.4.3 Finding a shared feature representation

At this point, one may notice that in order to apply optimal transport between $\widehat{\mathcal{D}}_{\mathcal{F}}^{T}$ and $\widehat{\mathcal{D}}_{\mathcal{F}}^{T}$, it is necessary to calculate the cost matrix $\boldsymbol{C}^f$ which is possible only if the numbers of source and target instances are equal. Furthermore, as source and target features are described by supposedly shifted distributions, aligning them directly using any arbitrary sets of instances may not be appropriate due to the differences in the representation spaces that may exist across the two domains. In order to tackle both of these problems, we propose to find a matching between $i$, the example index $\forall i = \{1, \dots, m\}$ describing the source features and $j$, the example index $\forall j = \{1, \dots, n\}$ describing the target features based on the original optimal transportation problem. More formally, based on the solution $\boldsymbol{\gamma}^*$ of the optimization problem given by Equation (4.1), we define the optimal subset of target instances $\boldsymbol{X}^{Tu}$ as:

$$\boldsymbol{X}^{Tu} = \{\boldsymbol{x}_j \in \boldsymbol{X}^T \mid j = \arg\max \gamma^*_{ij}, i \in \{1, \dots, m\}\}. \tag{4.9}$$

This particular choice of the algorithm **OT** rather than its regularized versions (**OT2** and **OT3**) is explained by the fact that we are interested in a sparse matching between the two sets, *i.e.*, the one limiting the spread of mass. We will give an empirical justification of using the **OT** algorithm for instance selection in the experimental section.

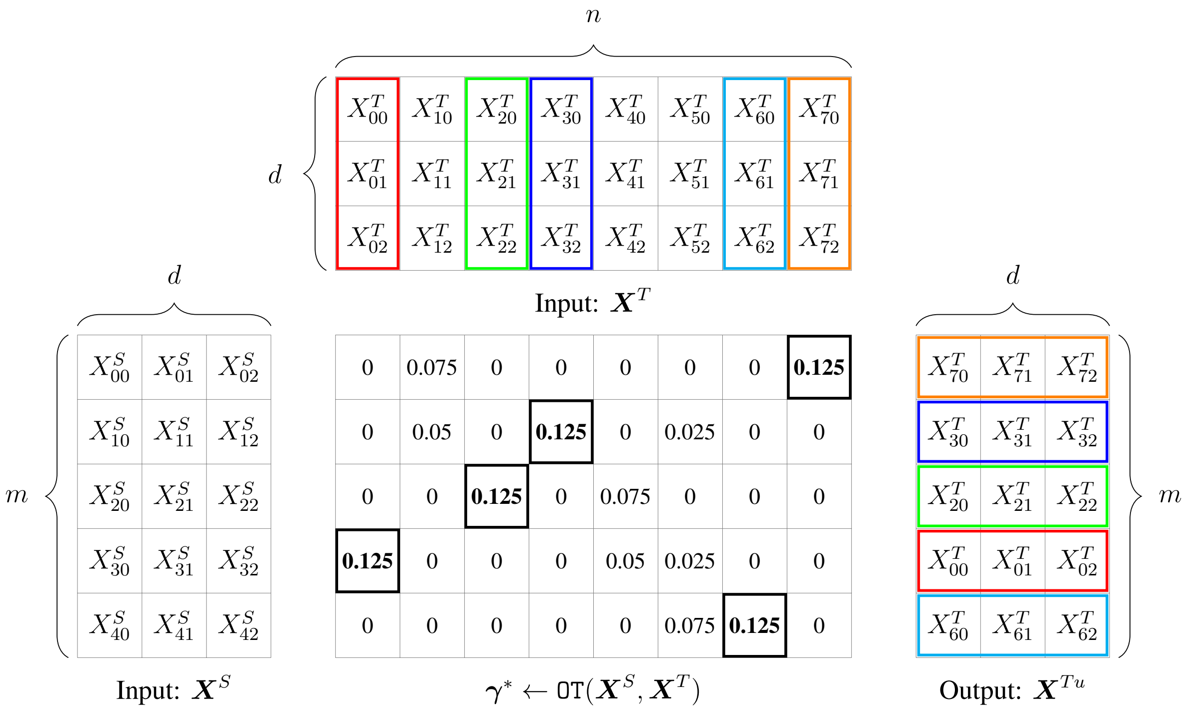

$n$
$d$
$X^T_{00}$ $X^T_{10}$ $X^T_{20}$ $X^T_{30}$ $X^T_{40}$ $X^T_{50}$ $X^T_{60}$ $X^T_{70}$
$X^T_{01}$ $X^T_{11}$ $X^T_{21}$ $X^T_{31}$ $X^T_{41}$ $X^T_{51}$ $X^T_{61}$ $X^T_{71}$
$X^T_{02}$ $X^T_{12}$ $X^T_{22}$ $X^T_{32}$ $X^T_{42}$ $X^T_{52}$ $X^T_{62}$ $X^T_{72}$
Input: $\boldsymbol{X}^T$
$d$
$m$
$X^S_{00}$ $X^S_{01}$ $X^S_{02}$
$X^S_{10}$ $X^S_{11}$ $X^S_{12}$
$X^S_{20}$ $X^S_{21}$ $X^S_{22}$
$X^S_{30}$ $X^S_{31}$ $X^S_{32}$
$X^S_{40}$ $X^S_{41}$ $X^S_{42}$
Input: $\boldsymbol{X}^S$
0 0.075 0 0 0 0 0 0.125
0 0.05 0 0.125 0 0.025 0 0
0 0 0.125 0 0.075 0 0 0
0.125 0 0 0 0.05 0.025 0 0
0 0 0 0 0 0.075 0.125 0
$\gamma^* \leftarrow \texttt{OT}(\boldsymbol{X}^S, \boldsymbol{X}^T)$
$d$
$m$
$X^T_{70}$ $X^T_{71}$ $X^T_{72}$
$X^T_{30}$ $X^T_{31}$ $X^T_{32}$
$X^T_{20}$ $X^T_{21}$ $X^T_{22}$
$X^T_{00}$ $X^T_{01}$ $X^T_{02}$
$X^T_{60}$ $X^T_{61}$ $X^T_{62}$
Output: $\boldsymbol{X}^{Tu}$

*Figure 4.2: Illustration of the example selection in the target domain described in Algorithm 4.1. For each source example, select the target example with which it has the highest coupling value.*

This process, summarized in Algorithm 4.1[2]and illustrated in Figure 4.2, is a required

[2] where `zscore(X)` is defined as follows: for each column of $X$, one subtracts its mean and divides by its

**Algorithm 4.1:** Example selection in target domain

```
Inputs  : $\boldsymbol{X}^S \in \mathbb{R}^{m\times d}$,
          $\boldsymbol{X}^T \in \mathbb{R}^{n\times d}$
Output : $\boldsymbol{X}^{Tu} \in \mathbb{R}^{m\times d}$ - optimal subset of target instances
  $\boldsymbol{X}^{\mathtt{S}}$=zscore($\boldsymbol{X}^{\mathtt{S}}$); $\boldsymbol{X}^{\mathtt{T}}$=zscore($\boldsymbol{X}^{\mathtt{T}}$)
  $\gamma^* \leftarrow$ OT($\boldsymbol{X}^{\mathtt{S}}, \boldsymbol{X}^{\mathtt{T}}$)
  $\boldsymbol{X}^{\mathtt{Tu}} \leftarrow \{\boldsymbol{x}_{\mathtt{j}} \in \boldsymbol{X}^{\mathtt{T}} | \mathtt{j} = \underset{\mathtt{i}\in\{\mathtt{1},\ldots,\mathtt{m}\}}{\text{argmax}}\, \gamma^*_{\mathtt{ij}}\}$
```

**Algorithm 4.2:** Feature ranking for domain adaptation

```
Inputs  : $\boldsymbol{X}^S \in \mathbb{R}^{m\times d}$,
          $\boldsymbol{X}^T \in \mathbb{R}^{n\times d}$
Output : List $F$ of $d$ most similar features from $\boldsymbol{X}^S$ and $\boldsymbol{X}^T$
  $\boldsymbol{X}^{\mathtt{Tu}} \leftarrow$ Algorithm4.1($\boldsymbol{X}^S, \boldsymbol{X}^T$)
  $\boldsymbol{X}^{\mathtt{S}\top}$=zscore($\boldsymbol{X}^{\mathtt{S}\top}$); $\boldsymbol{X}^{\mathtt{Tu}\top}$=zscore($\boldsymbol{X}^{\mathtt{Tu}\top}$)
  $\gamma^{*\mathtt{f}}$ = OT2($\boldsymbol{X}^{\mathtt{S}\top}, \boldsymbol{X}^{\mathtt{Tu}\top}, \lambda = 1$)
  F= argSortDesc($\{\gamma^{*\mathtt{f}}_{\mathtt{ii}} | \mathtt{i} \in \{\mathtt{1},\ldots,\mathtt{d}\}\}$)
```

preliminary step consisting in finding which examples will be used to describe the features in the source and target domains. The selection stage used to obtain $T_u$ relies on the intrinsic capacity of the coupling matrix to describe the probability of associating each source instance with each target instance based on their similarity.

### 4.4.4 Feature selection

Now, we let $\boldsymbol{X}^T = \boldsymbol{X}^{Tu}$ meaning that in Equation (4.8) the target features are described by the set $\boldsymbol{X}^{Tu}$ of target examples. If $n > m$, we invert the roles of $\boldsymbol{X}^S$ and $\boldsymbol{X}^T$ in Algorithm 4.1 and instead let $\boldsymbol{X}^S = \boldsymbol{X}^{Su}$. Furthermore, in a highly imbalanced classification setting, or in the presence of a large number of instances, we advise to first select a subset of source instances by balancing the examples according to their classes before applying Algorithm 4.1. This selection allows to capture a class information from the source domain without needing labeled examples from the target domain, and thus is still unsupervised *w.r.t.* the target domain.

We now solve the problem given in Equation (4.8) and obtain the optimal coupling $\boldsymbol{\gamma}^{*f} \in \mathbb{R}^{d\times d}$. Similar to what we have done at the example selection step, we analyze the values of the coupling matrix in order to determine the less shifted features across the two domains. The important difference, however, is that we sort the features by analyzing only the diagonal of the coupling matrix. This peculiarity is explained by the fact that the values on the diagonal correspond to the similarities between the same features in the shared source and target representation space. By transporting the features with the **OT2** algorithm, each source feature is transported to its

standard deviation.

nearest target features. Because of this, if a given feature is shifted across the two domains, then its mass will be uniformly spread on the target features so that its mass on the corresponding target feature will be rather small. Similarly, if a feature is similar between the source and target domains, then the majority of the mass of this source feature should be found on its corresponding target feature.

Based on this idea, we propose to construct the ordered list of features $F$, where the feature number $i$ in $F$ is the one having the $i^{\text{th}}$ highest coupling value on the diagonal of the coupling matrix, *i.e.*,

$$F = \arg\text{sort}(\{\gamma_{ii}^{*f} \mid i \in \{1, \dots, d\}). \tag{4.10}$$

By varying the parameter $\lambda$ in **OT2**, we can spread the mass of a source feature more or less uniformly when transporting it to the target features. Even though one may obtain different coupling values for different values of $\lambda$, it does not affect the order of features returned in $F$ allowing us to fix $\lambda = 1$ in all empirical evaluations to avoid hyper-parameter tuning.

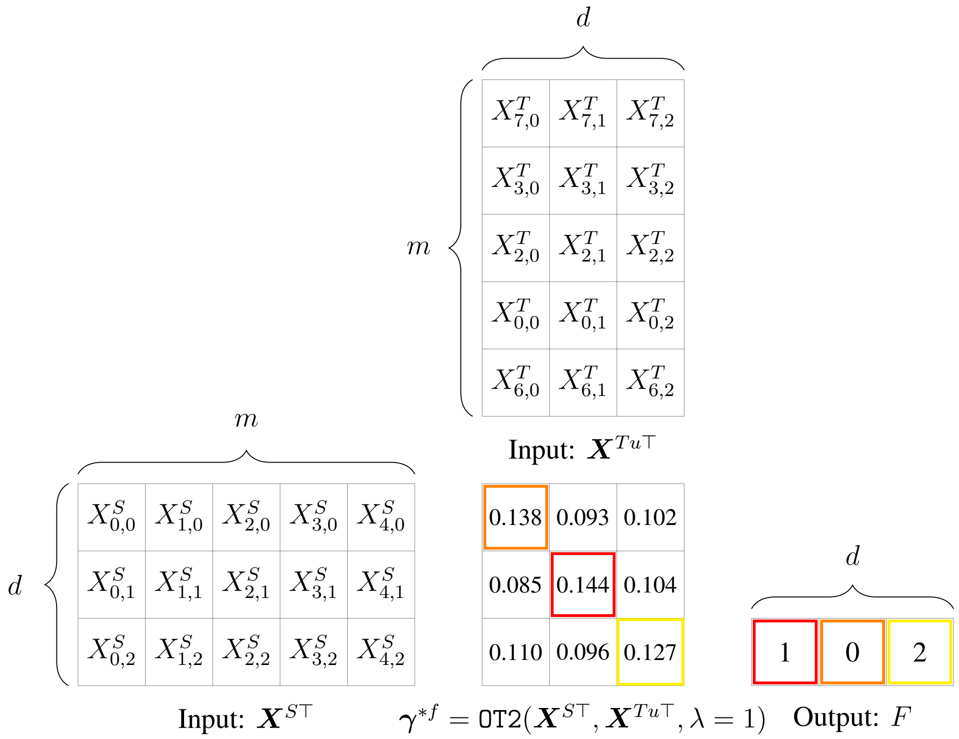

d
m
$X^T_{7,0}$ $X^T_{7,1}$ $X^T_{7,2}$
$X^T_{3,0}$ $X^T_{3,1}$ $X^T_{3,2}$
$X^T_{2,0}$ $X^T_{2,1}$ $X^T_{2,2}$
$X^T_{0,0}$ $X^T_{0,1}$ $X^T_{0,2}$
$X^T_{6,0}$ $X^T_{6,1}$ $X^T_{6,2}$
Input: $\boldsymbol{X}^{Tu\top}$
m
d
$X^S_{0,0}$ $X^S_{1,0}$ $X^S_{2,0}$ $X^S_{3,0}$ $X^S_{4,0}$
$X^S_{0,1}$ $X^S_{1,1}$ $X^S_{2,1}$ $X^S_{3,1}$ $X^S_{4,1}$
$X^S_{0,2}$ $X^S_{1,2}$ $X^S_{2,2}$ $X^S_{3,2}$ $X^S_{4,2}$
0.138 0.093 0.102
0.085 0.144 0.104
0.110 0.096 0.127
d
1 0 2
Input: $\boldsymbol{X}^{S\top}$
$\boldsymbol{\gamma}^{*f} = \texttt{OT2}(\boldsymbol{X}^{S\top}, \boldsymbol{X}^{Tu\top}, \lambda = 1)$
Output: $F$

*Figure 4.3: Illustration of our feature ranking for domain adaptation described in Algorithm 4.2. It consists in sorting the features by descending order of their coupling value across the two domains.*

The pseudo-code given in Algorithm 4.2 and illustrated in Figure 4.3 summarizes our feature selection method. After having obtained the ordered list of features $F$, we can use its $d^* < d$ first features for the classification problem at hand. It is worth noting that the proposed method can be applied as a pre-processing before using any domain adaptation algorithm to discard the features that are completely different across the two domains. On the other hand, it can also be applied in the "no adaptation setting" to select the common features between the training and test data.

## 4.5 Experimental evaluation

In this section, we provide an empirical study of the proposed algorithm based on three domain adaptation benchmarks which are Office/Caltech [Saenko et al., 2010, Gopalan et al., 2011], MNIST/USPS [Courty et al., 2017] and Amazon review [Germain et al., 2020], and on a clinical imaging database [Niaf et al., 2012] for computer-aided diagnostic task. The optimal transport algorithms **OT** used in Algorithm 4.1, **OT2** used in Algorithm 4.2 and **OT3** are available in the Python POT library[3], making our method straightforward to implement. Nevertheless, we make the Python implementation and the data used in our experiments (except the medical dataset for privacy reasons) publicly available[4] for the sake of reproducibility.

### 4.5.1 Experiments on visual domain adaptation data

The main assumption of our method is that not all features are equally useful for adapting a classifier from the source domain to the target one. This is especially the case for datasets described by features calculated using the Bag-of-Words (BoW) methods, such as, for instance, the features of the Office [Saenko et al., 2010]/Caltech [Gopalan et al., 2011] dataset.

**Office/Caltech dataset** For this dataset, the classification task is to assign an image to a class based on its content. It is composed of 4 domains: Amazon ($A$), Caltech ($C$), Webcam ($W$) and DSLR ($D$) containing $m = 958$, $m = 1123$, $m = 295$ and $m = 157$ images, respectively belonging each to one of $c = 10$ different classes (see Figure 4.4 for examples of images that can be found in the four domains). These domains form 12 pairs of domain adaptation sub-problems.

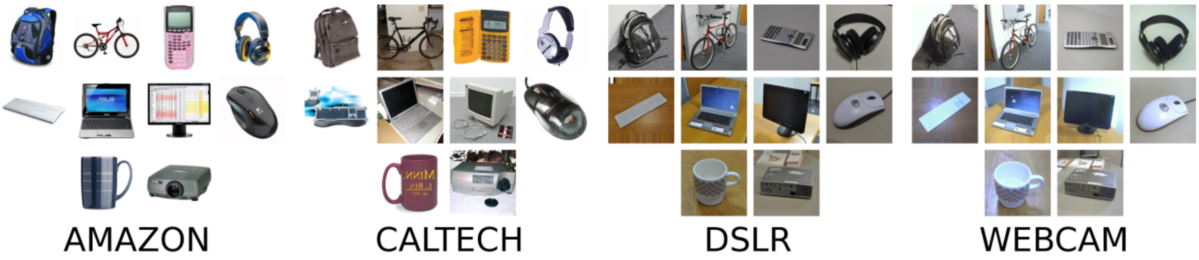


*Figure 4.4: Examples of images from the 10 classes in the four domains of the Office/Caltech dataset.*

In what follows, we use three different types of features: (1) SURF features [Bay et al., 2006] of size $d = 800$ constructed using the BoW method; (2) CaffeNet features [Jia et al., 2014] that

[3] `https://github.com/rflamary/POT`
[4] `https://leogautheron.github.io`

*Table 4.1: Classification accuracies in % and standard deviation with no adaptation for SURF, CaffetNet and GoogleNet features. Here, ↘ X (resp. ↗) indicates the use of the first X features sorted by decreasing (resp. ascending) similarity computed with Algorithm 4.2.*

| | SURF features | | | CaffeNet features | | | GoogleNet features | | |
|---|---|---|---|---|---|---|---|---|---|
| DA pairs | ↘400 | ↗400 | 800 | ↘ 512 | ↗ 512 | 4096 | ↘256 | ↗256 | 1024 |
| A→C | **25.4±2.4** | 15.4±1.5 | 23.1±1.6 | **74.9±2.0** | 29.8±2.4 | 71.7±3.5 | **85.7±1.2** | 64.7±2.4 | 84.6±1.1 |
| A→D | **24.5±2.9** | 16.2±3.0 | 21.9±2.4 | **78.8±3.5** | 20.4±2.8 | 76.0±3.5 | 86.7±2.4 | 68.6±4.9 | **88.4±2.5** |
| A→W | **27.5±2.2** | 16.2±2.6 | 26.0±2.1 | **77.6±1.9** | 20.2±3.5 | 66.0±4.6 | **85.4±3.1** | 51.8±5.9 | 83.5±2.8 |
| C→A | **24.8±1.4** | 14.1±2.2 | 21.2±2.4 | **83.7±1.8** | 38.7±4.5 | 82.1±2.2 | 90.4±1.2 | 74.5±3.4 | **90.6±1.7** |
| C→D | **25.5±3.7** | 15.5±2.8 | 22.8±3.6 | **76.2±3.6** | 24.1±3.4 | 74.2±4.9 | 88.3±2.7 | 68.5±4.3 | **88.6±2.7** |
| C→W | **23.3±3.0** | 13.9±2.2 | 20.6±3.5 | **75.4±3.5** | 20.3±3.2 | 70.3±5.3 | **86.2±2.7** | 54.3±4.6 | 83.3±2.4 |
| D→A | 25.7±2.0 | 15.8±2.9 | **26.7±1.7** | **75.4±2.1** | 20.8±3.8 | 68.7±2.9 | **84.2±2.0** | 46.4±4.2 | 82.3±1.6 |
| D→C | 23.8±1.9 | 16.0±2.1 | **24.8±1.5** | 65.0±2.6 | 21.5±2.5 | **66.6±1.8** | **80.5±1.7** | 46.9±2.8 | 77.8±2.5 |
| D→W | **53.6±3.5** | 22.1±3.4 | 53.3±2.7 | **92.6±2.0** | 32.8±5.1 | 91.9±1.9 | 96.5±1.1 | 81.5±3.4 | **97.4±0.8** |
| W→A | **23.7±1.9** | 15.6±1.9 | 23.1±1.5 | **81.5±1.2** | 18.8±2.4 | 68.3±3.0 | **89.7±0.8** | 55.8±2.5 | 87.0±1.2 |
| W→C | 18.1±1.7 | 12.0±1.6 | **19.5±1.0** | **72.2±1.1** | 23.4±2.1 | 61.2±2.1 | **83.7±1.1** | 49.9±2.5 | 79.4±1.2 |
| W→D | **63.4±3.6** | 21.7±3.4 | 52.4±2.6 | **96.5±1.5** | 49.7±3.2 | 96.3±1.0 | 98.9±0.8 | 93.4±1.8 | **99.2±0.5** |
| Mean | **29.9±2.5** | 16.2±2.5 | 27.9±2.2 | **79.2±2.2** | 26.7±3.3 | 74.4±3.0 | **88.0±1.7** | 63.0±3.5 | 86.8±1.8 |

are obtained by feeding the images to a pre-trained neural network based on the prominent AlexNet [Krizhevsky et al., 2012]; (3) GoogleNet features [Szegedy et al., 2015] obtained in the way identical to CaffeNet features using GoogleNet network. In order to obtain the CaffeNet and GoogleNet features, these two neural networks were first trained on ImageNet, a large dataset containing millions of images distributed across $c = 1000$ different classes. We removed their classification layer of size 1000 to use the output of the previous layer, giving $d = 4096$ features for CaffeNet and $d = 1024$ features for GoogleNet. We downloaded the pre-trained networks from the Caffe website [Jia et al., 2014] before using them to extract the features from our images, and this without doing any fine-tuning or any other modification of the networks apart from removing their last layer.

The experimental protocol used to evaluate the proposed method is based on the one presented by Courty et al. [2014]. For each adaptation (source, target) pair $S \to T$, we randomly sample 20 images per class (8 if $S$ is $D$). This gives us 200 images (resp. 80) for $S$. All images from $T$ are considered. We then apply Algorithm 4.2 with $\boldsymbol{X}^S$ and $\boldsymbol{X}^T$ to obtain the ordered list of features $F$. For an increasing number of features $d$, we use the first $d$ features of $F$ to, first adapt $S$ to $T$, and then use a 1-nearest neighbor classifier with the source adapted data as training set to compute the classification accuracy on the target data. We repeat this 19 times and report mean accuracies for each pair.

**Classification results** The classification results for the three types of features are given in Table 4.1. From this table, we see that by selecting 512 CaffeNet features having the highest similarity between the source and target domains, we obtain a mean accuracy of 79.2% across the 12 adaptation pairs compared to 74.4% accuracy obtained using all 4096 features. This

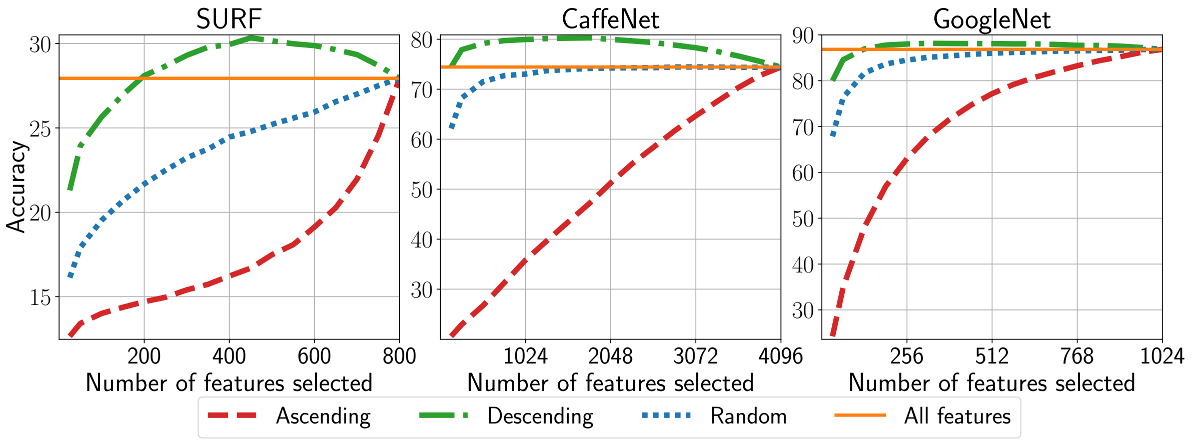


*Figure 4.5: Mean accuracies over the 12 DA pairs (from Table 4.1) without using an adaptation algorithm as a function of the number of features selected. Our method corresponds to the 'Descending' curve consisting in selecting the features ordered by decreasing similarity between source and target domains.*

*Table 4.2: Mean accuracies over the 12 DA pairs without applying adaptation using 3 different type of features: SURF (d=800), CaffeNet (d=4096) and GoogleNet (d=1024).*

| #features | SURF | CaffeNet | GoogleNet |
|---|---|---|---|
| ↘ $d/32$ | 21.3±2.4 | 74.4±2.9 | 80.0±2.6 |
| ↗ $d/32$ | 12.7±2.0 | 20.6±3.0 | 24.2±3.3 |
| ↘ $d/8$ | 25.7±2.6 | 79.2±2.2 | 86.9±1.8 |
| ↗ $d/8$ | 14.0±2.2 | 26.7±3.3 | 48.1±3.9 |
| ↘ $d/2$ | 29.9±2.5 | 80.0±2.2 | 88.1±1.8 |
| ↗ $d/2$ | 16.2±2.5 | 51.3±4.4 | 77.2±2.6 |
| d | 27.9±2.2 | 74.4±3.0 | 86.8±1.8 |

behaviour is further confirmed in Figure 4.5 (middle) that illustrates the obtained classification results for a number of features varying between 128 and 4096. We note that our method outperforms random feature selection while selecting the least similar features gives worse performances in all cases. We observe the same behavior for the SURF and GoogleNet features: our method gives better or almost identical performances on almost all domain adaptation pairs with significantly less features used. This confirms our claim about the efficiency of our proposed method for domain adaptation.

The general comparison of CaffeNet, SURF and GoogleNet features is given in Table 4.2. As before, we observe an important difference between taking the first most similar and dissimilar features across the two domains and better performances are obtained by taking a reduced number of features. Another noticeable point is that the performances of the SURF features

*Table 4.3: The arrays give the recognition accuracies in % and standard deviation with adaptation using the* ***OT3*** *algorithm for SURF, CaffetNet and GoogleNet features.*

| | SURF features | | | CaffeNet features | | | GoogleNet features | | |
|---|---|---|---|---|---|---|---|---|---|
| DA pairs | ↘400 | ↗400 | 800 | ↘2048 | ↗ 2048 | 4096 | ↘512 | ↗512 | 1024 |
| A→C | 27.2±1.6 | 28.5±2.2 | **30.3±1.5** | 82.6±1.1 | 73.0±2.0 | **82.7±0.7** | 89.5±0.7 | 82.5±1.8 | **89.7±0.8** |
| A→D | 39.0±4.3 | 32.9±3.4 | **40.9±2.6** | 91.3±1.5 | 85.6±3.0 | **93.3±1.3** | 91.4±0.9 | 94.8±1.6 | **93.5±0.4** |
| A→W | 34.1±2.4 | 30.5±2.9 | **34.4±2.0** | **94.8±1.0** | 74.1±4.1 | 92.1±1.1 | **96.6±1.3** | 88.6±3.0 | 95.8±1.1 |
| C→A | 34.2±2.4 | 30.9±3.7 | **36.9±2.6** | 89.0±1.4 | 84.1±1.7 | **89.2±0.8** | 92.5±1.0 | 89.3±1.7 | **93.8±0.5** |
| C→D | 43.4±5.5 | 38.3±4.0 | **44.2±5.0** | 90.4±1.2 | 89.0±2.3 | **93.3±1.2** | 91.8±0.8 | 94.6±1.5 | **93.9±0.9** |
| C→W | **38.3±4.8** | 30.7±4.2 | 37.9±5.3 | **94.0±1.3** | 75.7±3.6 | 90.5±1.9 | 96.0±1.1 | 90.2±2.4 | **96.8±0.7** |
| D→A | 27.1±2.4 | 24.9±2.7 | **28.8±1.7** | 86.0±1.8 | 75.5±4.0 | **86.6±1.4** | 90.2±1.5 | 82.9±3.1 | **91.1±1.1** |
| D→C | 27.8±1.3 | 26.9±2.6 | **28.9±1.4** | 77.7±3.4 | 73.6±2.8 | **80.0±3.0** | 86.3±1.6 | 81.5±2.0 | **89.1±0.8** |
| D→W | 66.5±2.9 | 54.7±2.9 | **68.6±2.1** | **98.1±0.7** | 92.4±1.3 | 96.6±0.6 | **98.2±0.8** | 96.0±0.9 | 97.9±0.6 |
| W→A | 35.6±1.0 | 23.6±3.4 | **37.5±0.8** | **87.7±1.2** | 71.1±1.9 | 86.1±1.9 | 92.8±0.4 | 85.5±2.1 | **92.9±0.3** |
| W→C | 31.5±1.2 | 29.3±2.4 | **34.3±1.1** | **78.8±1.8** | 67.0±1.9 | 78.1±1.8 | 89.4±1.4 | 82.5±1.5 | **90.3±1.0** |
| W→D | **71.8±2.0** | 57.9±1.4 | 71.4±1.7 | 95.4±1.2 | 96.7±1.0 | **97.3±0.7** | 95.7±1.1 | **99.6±0.9** | 97.2±1.2 |
| Mean | 39.7±2.6 | 34.1±3.0 | **41.2±2.3** | **88.8±1.5** | 79.8±2.5 | **88.8±1.4** | 92.5±1.0 | 89.0±1.9 | **93.5±0.8** |

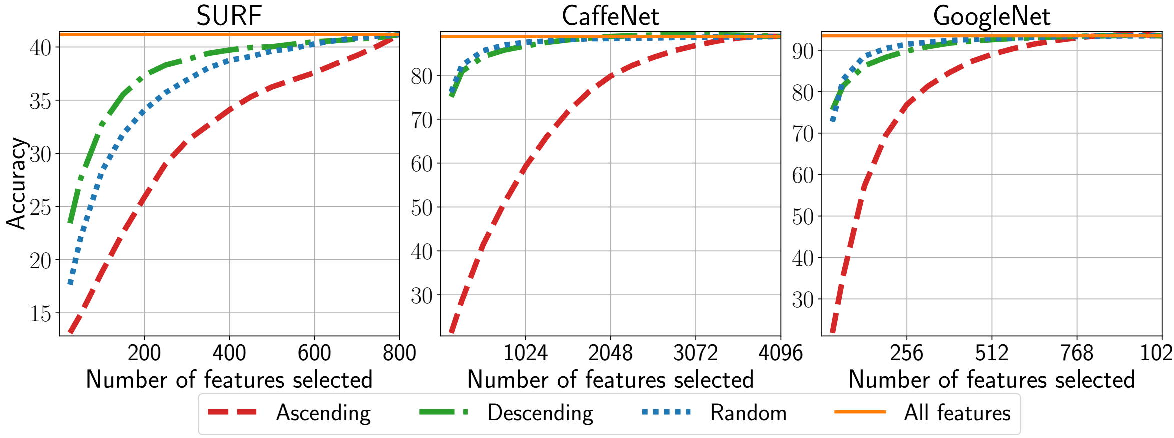


*Figure 4.6: Mean accuracies over the 12 DA pairs using the* ***OT3*** *adaptation algorithm as a function of the number of features selected.*

are far behind the CaffeNet features, the latter being slightly worse than GoogleNet features. Even by taking a small number (1024/32 = 32) of GoogleNet features, we obtain a mean accuracy of 80.0% which is at least as good as all the other configurations using SURF and CaffeNet features. To summarize, the presented results clearly show that the order of features returned by our method is directly correlated with their adaptation capacities.

We saw in the previous experiment that our method works for different types of features without applying any adaptation algorithm. We present in Table 4.3 and Figure 4.6 the impact of using an adaptation algorithm that takes as input a reduced set of features returned by our method. Several important conclusions can be made based on these results. First, we notice that our algorithm does not improve the classification results compared to the performance of the **OT3** algorithm with a randomly selected subset of the CaffeNet and GoogleNet features.

*Table 4.4: Mean recognition accuracies in %, standard deviation and sum of total computational time (over the 12 DA pairs and 19 iterations) in seconds for different adaptation algorithms using the CaffeNet features.*

| Method | ↘512 | | ↘1024 | | ↘2048 | | 4096 | |
|---|---|---|---|---|---|---|---|---|
| No adapt. | 79.2±2.2 | 0.00s | 79.9±2.3 | 0.00s | 80.0±2.2 | 0.00s | 74.4±3.0 | 0.00s |
| **CORAL** | 80.5±1.8 | 110.43s | 80.8±1.9 | 587.69s | 80.4±1.7 | 3996.20s | 80.1±1.7 | 29930.39s |
| **SA** | 81.8±2.0 | 13.25s | 82.5±1.8 | 32.09s | 82.9±1.7 | 66.71s | 83.0±1.7 | 169.71s |
| **TCA** | 83.5±2.2 | 221.08s | 85.0±1.9 | 223.62s | 85.8±1.8 | 229.48s | 85.9±1.7 | 242.71s |
| **OT3** | 84.2±2.4 | 19.50s | 86.7±1.9 | 31.76s | 88.8±1.5 | 54.07s | 88.8±1.4 | 97.47s |

As explained in the introduction, **OT3** finds a new latent projection of the source data in order to leverage the shift between the two domains. In this case, eliminating shifted features does not directly contribute to an improved classification performance as **OT3** algorithm can handle the reduction of shift between the two domains pretty well on its own. However, we can also observe that the performance of **OT3** with a reduced "Ascending" set of features reaches its maximal value sooner than when no adaptation is performed in (4.5). This is explained by the fact that **OT3** successfully adapts the most shifted features. Moreover, it is important to notice that with or without adaptation, the curve "Ascending" is far below the other curves. This means that the features identified by our proposed method as most dissimilar between the two domains are harmful on the performances of the final classifier if used alone.

It is quite intuitive to assume that by selecting a subset of features, we decrease the computational complexity of the adaptation and classification algorithms that are used later. To support this claim, we present below an additional study of the impact of reducing the number of features on both the computational time and classification performance for several adaptation algorithms below.

**Running time speed-up** For this experiment, we evaluated the gain in computational time of different adaptation algorithms as a function of the number of features selected by our method. To this end, we compared the "no adaptation" setting with four state-of-the-art adaptation algorithms: **CORAL** [Sun et al., 2016], **SA** [Fernando et al., 2013], **TCA** [Pan et al., 2010] and **OT3** [Courty et al., 2014]. We fixed the subspace dimensions of **SA** and **TCA** to 80 (or to the number of features selected when smaller than 80) while for **OT3** we set $\lambda = 2$ and $\eta = 1$. Even if from Table 4.2 we obtained the best performances with GoogleNet features, we select for this experiment the CaffeNet features to better see the computational gain because they have the largest dimensionality (4096).

The results of this evaluation are presented in Table 4.4. From these results, we see that by selecting 2048 out of 4096 most similar features, we are able to obtain slightly better classification performances for all adaptation methods compared to the case when all features are used. Moreover, the computation time required by the algorithms greatly decreases. When only

512 features are used, an even more impressive speed up is obtained with a very slight drop in performance for the last three methods. These results confirm that our method is capable of finding subsets of similar features between source and target domains that can give comparable and sometimes even improved classification performances while decreasing considerably the computation time required for adaptation methods to converge.

**Comparison of different instance selection strategies** As explained in Section 4.4.3, our method requires to select a set of examples that describe the features in the source and target domains before computing the similarity between them. For our method, we propose to select these examples using the **OT** algorithm between the source and target examples. In Table 4.5, we evaluate two other example selection methods on the Office/Caltech dataset: the first one is based on the random selection of the examples while the second uses a 1-Nearest-Neighbor (1NN) algorithm instead of the **OT** method. The computation of the features' rank is then done in the same way as that of presented in Algorithm 4.2.

*Table 4.5: Mean accuracies over the 12 adaptation pairs without applying adaptation on CaffeNet features obtained using different example selection methods. Our proposed selection method is **OT**.*

| #features | Random | **OT** | 1NN |
|---|---|---|---|
| ↘128 | 42.7±6.0 | **74.6±3.4** | 72.8±2.9 |
| ↗128 | 43.9±5.6 | 20.8±2.7 | 22.3±3.1 |
| ↘512 | 68.1±4.5 | **79.3±2.6** | 79.1±2.7 |
| ↗512 | 60.8±5.3 | 27.1±3.3 | 27.6±3.4 |
| ↘2048 | 75.9±3.2 | **80.1±2.2** | 79.6±2.7 |
| ↗2048 | 68.9±4.8 | 52.3±4.2 | 50.4±4.4 |
| 4096 | 75.2±3.0 | 75.2±3.0 | 75.2±3.0 |

From this table, we can see that a random selection of instances gives poor results for different numbers of features considered in our study. On the other hand, we observe that both **OT** and the 1NN algorithm provide close performances in identifying similar and dissimilar features with a slight superiority of the optimal transport based method. In order to discriminate between the two, we demonstrate in Figure 4.7 the pitfalls of the 1NN based selection that can occur when the vast majority of source points are associated with a handful of target instances. We can see that for the two considered toy datasets, the selection of target instances based on the 1NN algorithm leads to a distribution that does not reflect the true distribution of the target data. If we would have selected points in the target domain randomly, we would still have the same target distribution, but as we have shown previously in Table 4.5, the random selection gives worse classification performances. On the other hand, the proposed

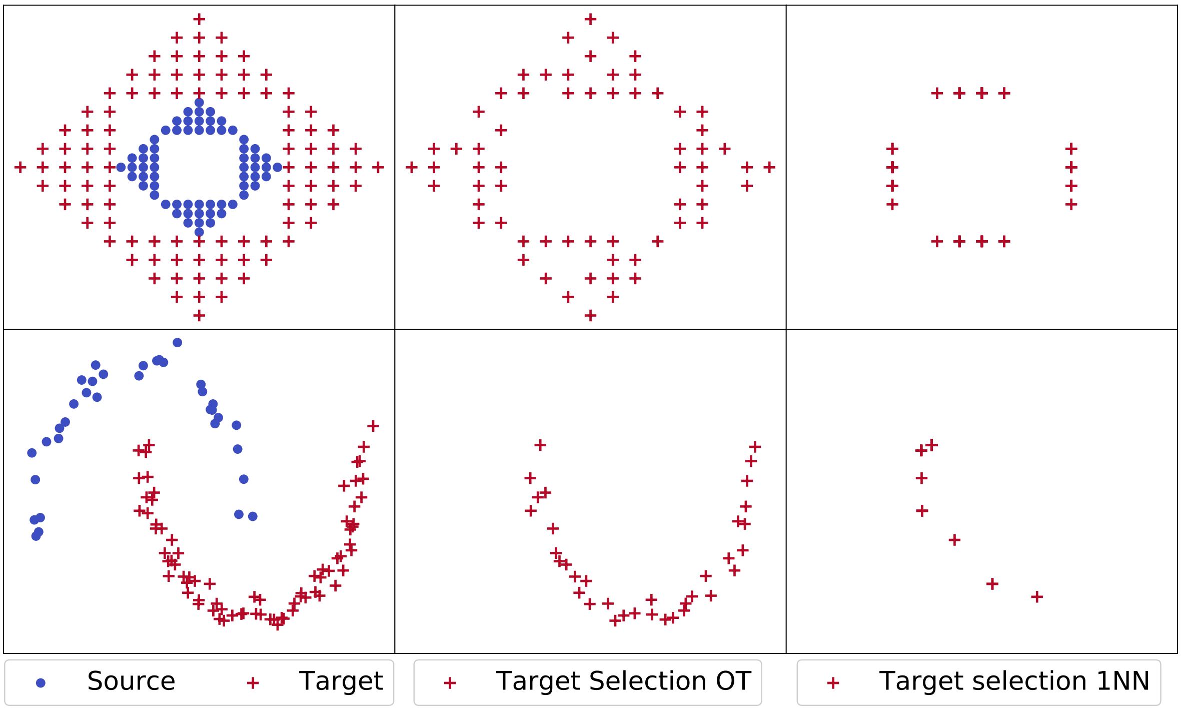


*Figure 4.7: Two toy examples where we generated a source and a target distribution (left) before using the example selection procedure in the target domain using the **OT** algorithm (in the middle) and the 1NN selection (on the right).*

strategy for the selection of target examples through the **OT** algorithm allows to obtain both good classification performances and to preserve the target data distribution. Recall that he computation of the **OT** has a squared space complexity compared to the linear complexity of the 1NN selection. Consequently, the use of the example selection with the 1NN algorithm can present a good alternative for large-scale machine learning problems.

### 4.5.2 Experiments on digit recognition and textual product reviews

**Description of the digit recognition datasets** We use in this experiment two digits datasets: MNIST and USPS, where the task is to assign to each image the digit (among the $c = 10$ labels) drawn on the image. MNIST is composed of $m = 70000$ gray-scale images of size 28 by 28 giving a total of $d = 784$ features taking their value between 0 (white pixel) and 255 (black pixel). USPS has $m = 9298$ gray-scale images of size 16 by 16 giving $d = 256$ features of values between $-1$ (white pixel) and $+1$ (black pixel). As pre-processing, we resize the MNIST images from 28 by 28 pixels to 16 by 16 pixels, and we normalize the pixel values in both datasets to be between 0 (white pixel) and $+1$ (black pixel). We provide in Figure 4.8 some example of images from the MNIST and USPS datasets after pre-processing. Both datasets are well balanced and each class represents between 8% and 16% of all the examples in the dataset. These two datasets form 2 pairs of domain adaptation sub-problems.

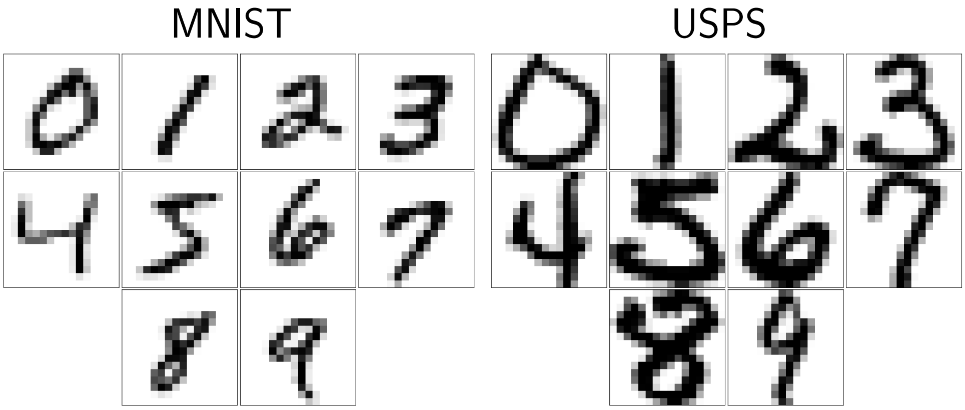


*Figure 4.8: Examples of images for the* 10 *digit classes in the MNIST and USPS datasets.*

**Description of the textual product review dataset** The amazon review dataset contains reviews of four categories of products: $m = 6465$ reviews of books, $m = 5586$ reviews of DVDs, $m = 7681$ reviews of electronic and $m = 7945$ reviews of kitchens. We use the pre-processed data provided by Germain et al. [2020] where the task is to distinguish between positive and negative reviews. The datasets are all balanced with approximately as many positive as negative reviews. The pre-processing of the data consists in a conversion from the original textual representation to a numerical one. To this aim, Germain et al. [2020] first established a list of all possible uni-grams (one character) and bi-grams (two adjacent characters) present in all reviews. Only the $d = 5000$ more common uni-grams and bi-grams among all reviews are kept, and each review is described by the number of occurrences of each gram inside the review. These four datasets form 12 domain adaptation pairs.

**Classification results** We report the classification results of our method using a 3NN algorithm for the digit recognition benchmark, and a linear SVM for the Amazon review benchmark. The mean results over all the domain adaptation pairs are depicted in Figure 4.9 and the detailed results for each domain adaptation pair are described in Table 4.6. For the digit recognition task, our method allows to improve the performances by selecting a subset of features when learning from USPS and testing on MNIST while showing a slight loss of performances from MNIST to USPS. For the amazon review problem, we do not see any improvement of the performances by selecting the most similar features compared to using all the features. However, in both benchmarks we observe the same behavior as with the previous Office/Caltech benchmark: selecting the most dissimilar features between the two domains (curves “Ascending”) gives much worst performances than selecting randomly a subset of features. This confirms that our method can successfully identify dissimilar features between the two domains that may degrade the performances of the final classifier.

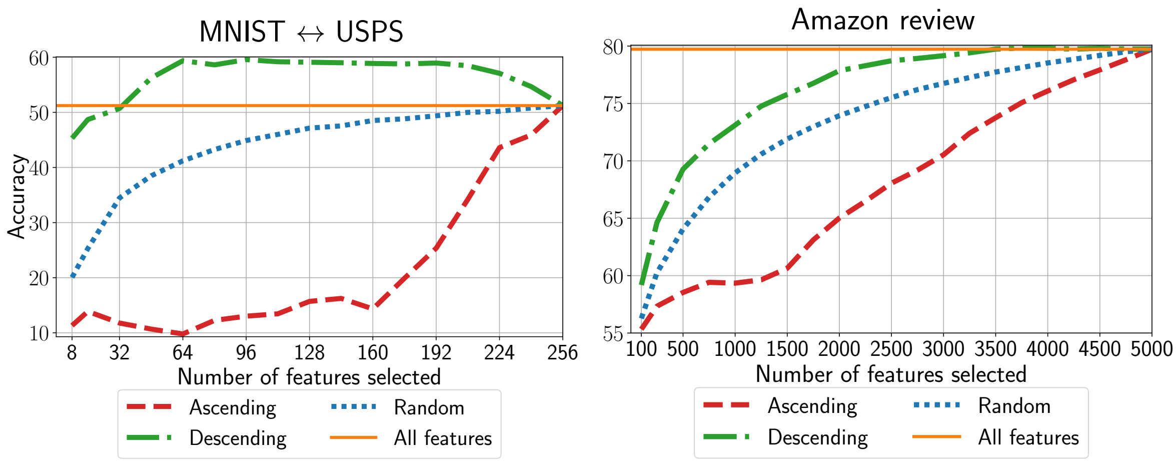


*Figure 4.9: Mean accuracies over the domain adaptation pairs on the digit recognition and amazon review datasets.*

*Table 4.6: Recognition accuracies for the digit recognition benchmark (upper part of the array) and the Amazon review benchmark (bottom part of the array).*

| DA pairs | ↘ 64 | ↗ 64 | 256 |
|---|---|---|---|
| M→U | 67.6 | 12.8 | 70.8 |
| U→M | 51.1 | 6.8 | 31.7 |
| Mean | 59.4 | 9.8 | 51.2 |
| DA pairs | ↘ 2500 | ↗ 2500 | 5000 |
| B→D | 79.8 | 72.5 | 81.5 |
| B→E | 76.2 | 64.3 | 76.1 |
| B→K | 77.8 | 65.9 | 79.2 |
| D→B | 80.3 | 72.4 | 82.5 |
| D→E | 77.8 | 66.4 | 78.8 |
| D→K | 79.5 | 66.2 | 80.1 |
| E→B | 73.5 | 63.4 | 74.3 |
| E→D | 74.2 | 65.0 | 75.5 |
| E→K | 88.0 | 74.9 | 89.1 |
| K→B | 74.6 | 63.9 | 75.2 |
| K→D | 76.6 | 66.5 | 77.0 |
| K→E | 86.4 | 75.4 | 87.5 |
| Mean | 78.7 | 68.1 | 79.7 |

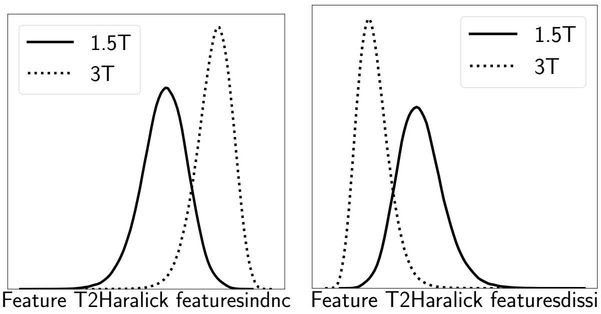


*Figure 4.10: Example of distribution of 2 features illustrating the shift between the source and target domains.*

### 4.5.3 Experiments on a medical imaging dataset

We now proceed to the evaluation of our method on a clinical dataset of multi-parametric magnetic resonance images (mp-MRI) collected to train a computer-aided diagnosis system for prostate cancer mapping [Niaf et al., 2012, Aljundi et al., 2015]. This system learns a binary decision model in a multidimensional feature space based on training examples (voxels) from different classes of interest. This model is then used to generate cancer probability maps.

**Data description** The considered database consists of 90 mp-MRI exams acquired with different imaging protocols on two different scanners (49 patients on a 1.5T scanner and 41 on a 3T scanner), thus producing heterogeneous datasets. Each individual voxel is described by a binary label (Cancer, Non Cancer) and a set of $d = 95$ handcrafted features consisting of image descriptors, texture coefficients, gradients and other visual characteristics (more details in the paper from Niaf et al. [2012]). Some of these 95 features have a clear shift between the two domains, as illustrated in Figure 4.10. The number of available instances in both domains is shown in Table 4.7. Our goal is to learn a classifier on annotated 1.5T voxels, representing the source domain, performing well on 3T voxels, considered as the target domain, without using labels from the latter one.

*Table 4.7: Distribution of the MRI voxels between the Cancer and Non Cancer classes in the source and target domains.*

| Class | #voxels 1.5T | #voxels 3T |
|---|---:|---:|
| Non cancer | 363,222 | 846,556 |
| Cancer | 56,126 | 140,840 |
| Total | 419,348 | 987,396 |

**Evaluation protocol** We first randomly sample a set $S$ of $m = 1500$ voxels equiproportionally from the 49 1.5T exams and both classes of interest. Then, we use Algorithm 4.1 on $S$ and

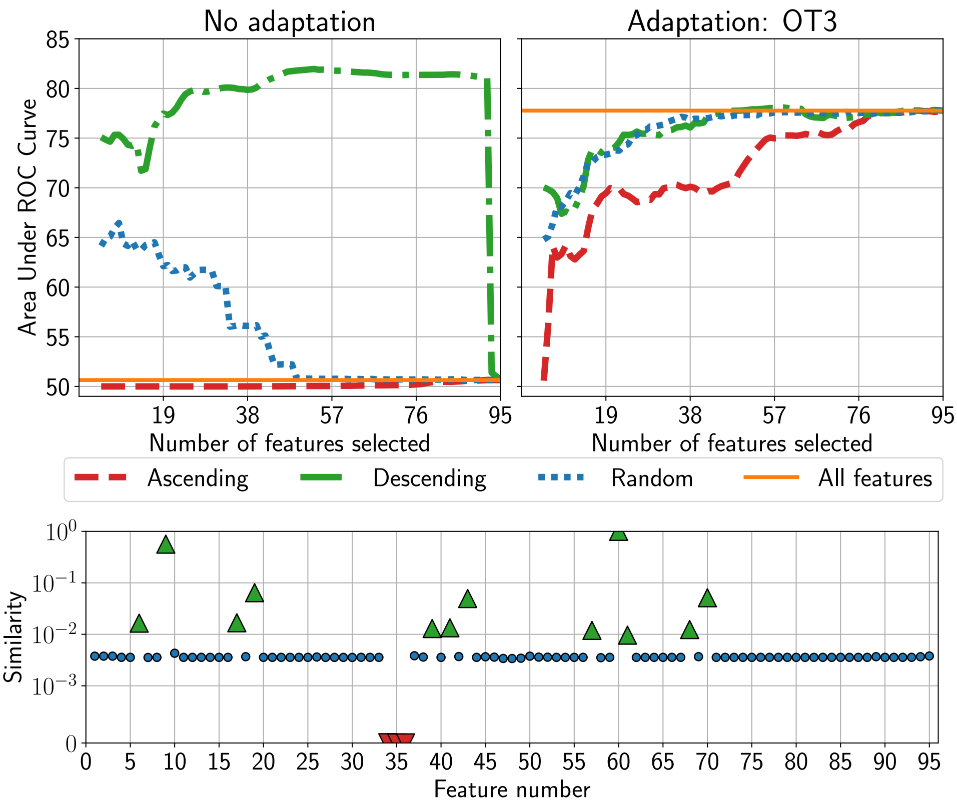


*Figure 4.11: Performance of our method on the clinical MRI database with no adaptation (top row, left) and using the **OT3** algorithm (top row, right). The log-scaled similarity of features across the two domains estimated by our algorithm is given in the bottom row. We observe that our method correctly identifies the three most shifted features that lead to an important drop in classifier's performance.*

on $T$ as $n = 20000$ randomly sampled voxels from the 41 3T exams to obtain $T_u$. This step is followed by the adaptation of $S$ to $T_u$, the training of a linear SVM on $S_a$ and a testing step on all voxels from the 3T target domain.

We used the area under the ROC Curve (AUC) defined in Section 1.1.2 as the diagnostic performance measure. This comes from the fact that both the source and the target domains exhibit an important class imbalance with 86% of non-cancer voxels. In this case, the classification accuracy used in the previous experiments does not provide a truthful picture of the classifier's performance. Our feature selection method is used as a standalone method and in combination with the **OT3** adaptation algorithm. As before, we repeat this process 20 times, and we report the mean AUC over the 20 iterations.

**Obtained results** The results for this dataset are shown in Figure 4.11. When all the 95 features are used, we obtain an AUC of 50% without adaptation, corresponding to the worst possible performance with no distinction between Cancer and Non cancer classes. By applying our feature selection algorithm (the "Descending" curve) in a standalone manner, we are able to reach an AUC of 80% with a significant drop in performance when the 3 most dissimilar features are added. On the other hand, using our feature selection algorithm before applying an adaptation algorithm reduces greatly the number of features needed to achieve comparable performances. This benefit presents an important computational gain when high-dimensional datasets are considered. Finally, we argued that one of the strengths of our method is its ability to identify the original features causing the shift between the source and target domains. To this end, we plot in Figure 4.11 the coupling values used to order the features by their similarity across the two domains. From this Figure, we can see that our algorithm allows to identify the three most shifted features that lead to a significant performance drop observed previously.

## 4.6 Conclusions and perspectives

In this chapter, we presented a new feature selection method for domain adaptation based on optimal transport. Building upon a recent theoretical work on optimal transport in domain adaptation, we proposed a feature selection method that transports the empirical distribution of features in the source domain to that of the target one in order to obtain a coupling matrix representing their joint distribution. This coupling matrix is further used to identify the subset of features that remain unshifted across the two domains. We evaluated our method on both benchmark and real-world datasets and showed its efficiency in identifying the subset of features that successfully reduces the discrepancy between the two domains. Furthermore, we illustrated the usefulness of our method in reducing the computational time of several state-of-the-art methods that converge faster when taking as input a reduced set of features returned by our algorithm.

The possible future investigations that may follow up the presented work are many. First of all, we would like to combine our feature selection algorithm with a feature-transformation domain adaptation algorithm in a way such that the projection of data and the selection of features would be performed simultaneously. The potential interest of this joint approach would be to reduce the computational complexity of the adaptation methods and to improve their performance while maintaining the ease of interpretability of the obtained results. On the other hand, it would be also very interesting to extend the proposed framework to the general transfer learning scenario where the source and target tasks are not necessarily the same. In this case, the feature selection algorithm would have to take into account the discriminative power of each source feature in the target domain. Solving this problem in an unsupervised setting is a very challenging task that would require an efficient feature expressiveness measure to be introduced. We believe that this future perspective would be of a great interest in many real-world applications, notably the health-care one, where the manual labeling of the produced MRI scans represents an important bottleneck due to its highly time-consuming nature.

# Conclusion and Perspectives

In this thesis, we tackled the problem of learning a suitable representation of the data in the specific context where the supervision on the data of interest is limited. We considered this setting in different possible scenarios *(i)* in Chapter 2, the training set is composed of a small number of positive labeled examples that often happens, *e.g.*, in anomaly detection tasks. To address this problem, we introduce a theoretically well rooted metric learning algorithm specifically dedicated to deal with highly imbalanced datasets *(ii)* in Chapter 3, we assume that the whole training set is composed of a limited number of labeled examples, preventing the use of deep neural network-based approaches. We tackle this problem by learning iteratively through boosting and kernel random Fourier features a model and a representation that capture complex decision boundaries and generalizes well with few examples and *(iii)* in Chapter 4, the source set is made of labeled examples but the target domain on which the model will be deployed does not provide labeled examples. We cope with this difficult setting by learning a joint representation between the source and target domains through the lens of a feature selection algorithm that optimizes an optimal transport problem to find the most similar features between the two domains. In each of these scenarios, we proposed an original contribution and showed how a good representation of the data can help to create classification models more suited than with the original representation of the data. Our contributions are mainly algorithmic, but each of them takes strong root in a theoretical analysis.

A common perspective to the different contributions lies in the scalability of the algorithms with respect to the number of training instances. This is especially true for the contributions in Chapters 2 and 4 that both rely on pairwise distance matrices between two potentially large sets of examples. In the metric learning part, the distance matrices are computed to ensure that all constraints are satisfied by the learned metric. A promising perspective would consist in incorporating the ideas of neural network methods and perform a batch optimization: at each iteration, only the constraints on a small set of examples would be checked, instead of all the examples as done at the moment. Even if the contribution random Fourier features-based algorithm presented in Chapter 3 is relatively fast compared to the state-of-the-art competitors, a batch optimization could also makes the computation much faster by computing the residuals for a reduced number of examples.

In our optimal transport-based feature selection, (4), a discrete transport map is computed between two sets of examples and provides the transport plan between any two points of the sets. Instead, a possibility could be to rely on optimal transport implementations that learn a continuous transport function, that could eventually also be learned using batch optimization.

A second perspective in line with what is done in Chapter 3 would be to provide a kernelized version of our metric learning algorithm **IML**. The use of random Fourier features seems appropriate here as like most metric learning algorithms, this method allows to produce a mapping of the point in a different representation space. A possibility in the setting of metric learning would consist in learning a linear weighted combinations of the random features to implicitly induce non linearities.

# List of Figures

# List of Tables

# List of Algorithms

# Bibliography

Charu C. Aggarwal. *Outlier Analysis*. Springer, 2013. (Cited on pages 34 and 51.)

Raj Agrawal, Trevor Campbell, Jonathan Huggins, and Tamara Broderick. Data-dependent compression of random features for large-scale kernel approximation. In *the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1822–1831, 2019. (Cited on pages 26, 60, and 67.)

Rahaf Aljundi, Jérôme Lehaire, Fabrice Prost-Boucle, Olivier Rouvière, and Carole Lartizien. Transfer learning for prostate cancer mapping based on multicentric MR imaging databases. In *Medical learning meets medical imaging workshop*, pages 74–82. Springer, 2015. (Cited on page 100.)

Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. Improved guarantees for learning via similarity functions. In *the 21st Annual Conference on Learning Theory (COLT)*, pages 287–298, 2008. (Cited on page 60.)

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research (JMLR)*, 3(Nov):463–482, 2002. (Cited on page 15.)

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision (ECCV)*, pages 404–417. Springer, 2006. (Cited on pages 1 and 91.)

Aurélien Bellet, Amaury Habrard, and Marc Sebban. *Metric learning*, volume 9. Morgan & Claypool Publishers, 2015. (Cited on pages 2, 3, 9, 19, 24, 32, 35, 39, 40, 41, 45, 46, 47, and 48.)

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems (NeurIPS)*, pages 137–144, 2007. (Cited on pages 2 and 82.)

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010. (Cited on page 82.)

Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006. (Cited on page 9.)

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992. (Cited on page 19.)

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research (JMLR)*, 2(Mar):499–526, 2002. (Cited on pages 3, 15, 16, 34, 39, 40, 41, and 46.)

Paula Branco, Luís Torgo, and Rita P. Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, 49(2):1–50, 2016. (Cited on page 34.)

Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and regression trees*. CRC press, 1984. (Cited on page 22.)

Qiong Cao, Zheng-Chu Guo, and Yiming Ying. Generalization bounds for metric and similarity learning. *Machine Learning*, 102(1):115–132, 2016. (Cited on page 35.)

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009. (Cited on pages 34 and 35.)

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011. (Cited on page 21.)

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and Philip W. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357, 2002. (Cited on pages 34, 51, and 52.)

Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. SMOTE-Boost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery*, pages 107–119. Springer, 2003. (Cited on page 34.)

Corinna Cortes and Vladimir N. Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995. (Cited on page 19.)

Nicolas Courty, Rémi Flamary, and Devis Tuia. Domain adaptation with regularized optimal transport. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 274–289. Springer, 2014. (Cited on pages 84, 85, 92, and 95.)

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, pages 1853–1865, 2017. (Cited on page 91.)

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967. (Cited on page 18.)

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems (NeurIPS)*, pages 2292–2300, 2013. (Cited on page 27.)

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. (Cited on pages 25, 32, 37, and 50.)

Georgios Douzas and Fernando Bacao. Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with applications*, 91:464–471, 2018. (Cited on page 34.)

Petros Drineas and Michael W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research (JMLR)*, 6:2153–2175, 2005. (Cited on page 60.)

Chris Drummond and Robert C. Holte. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer, 2003. (Cited on page 34.)

Sahibsingh A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 325–327, 1976. (Cited on page 19.)

Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence (IJCAI)*, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001. (Cited on page 34.)

Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36, 2004. (Cited on page 34.)

Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *Journal of machine learning research (JMLR)*, 6(Dec):1889–1918, 2005. (Cited on pages 3 and 20.)

Lin Feng, Huibing Wang, Bo Jin, Haohao Li, Mingliang Xue, and Le Wang. Learning a distance metric by balancing KL-divergence for imbalanced datasets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(12):2384–2395, 2018. (Cited on page 34.)

Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE international conference on computer vision*, pages 2960–2967, 2013. (Cited on page 95.)

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Efficient top rank optimization with gradient boosting for supervised anomaly detection. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 20–35. Springer, 2017. (Cited on page 34.)

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning (ICML)*, volume 96, pages 148–156. Citeseer, 1996. (Cited on page 13.)

Jerome H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. (Cited on pages 4, 13, 24, 29, 60, 63, 64, 65, and 109.)

Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011. (Cited on page 34.)

Léo Gautheron, Amaury Habrard, Emilie Morvant, and Marc Sebban. Apprentissage de métrique pour la classification supervisée de données déséquilibrées. In *Conférence sur l'Apprentissage automatique (CAp)*, 2018a. (Cited on pages 6 and 31.)

Léo Gautheron, Ievgen Redko, and Carole Lartizien. Feature selection for unsupervised domain adaptation using optimal transport. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2018b. (Cited on pages 5 and 81.)

Léo Gautheron, Pascal Germain, Amaury Habrard, Gaël Letarte, Emilie Morvant, Marc Sebban, and Valentina Zantedeschi. Revisite des "random Fourier features" basée sur l'apprentissage PAC-Bayésien via des points d'intérêts. In *Conférence sur l'Apprentissage automatique (CAp)*, 2019a. (Cited on pages 6 and 59.)

Léo Gautheron, Amaury Habrard, Emilie Morvant, and Marc Sebban. Metric learning from imbalanced data. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019b. (Cited on pages 5 and 31.)

Léo Gautheron, Pascal Germain, Amaury Habrard, Guillaume Metzler, Emilie Morvant, Marc Sebban, and Valentina Zantedeschi. Apprentissage d'ensemble basé sur des points de repère avec des caractéristiques de Fourier aléatoires et un renforcement du gradient. In *Conférence sur l'Apprentissage automatique (CAp)*, 2020a. (Cited on pages 5 and 59.)

Léo Gautheron, Pascal Germain, Amaury Habrard, Guillaume Metzler, Emilie Morvant, Marc Sebban, and Valentina Zantedeschi. Landmark-based ensemble learning with random Fourier features and gradient boosting. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2020b. (Cited on pages 5 and 59.)

Léo Gautheron, Emilie Morvant, Amaury Habrard, and Marc Sebban. Metric learning from imbalanced data with generalization guarantees. *Pattern Recognition Letters*, 133:298–304, 2020c. (Cited on pages 5 and 31.)

Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. PAC-Bayesian learning of linear classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 353–360, 2009. (Cited on pages 17 and 63.)

Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. PAC-Bayes and domain adaptation. *Neurocomputing*, 379:379–397, 2020. (Cited on pages 91 and 98.)

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT Press, 2016. (Cited on pages 2 and 23.)

Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *Proceedings of the 2011 International Conference on Computer Vision (ICCV)*, pages 999–1006, 2011. (Cited on page 91.)

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*, 13(Mar): 723–773, 2012. (Cited on page 87.)

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research (JMLR)*, 3(Mar):1157–1182, 2003. (Cited on page 83.)

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005. (Cited on page 34.)

Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009. (Cited on page 34.)

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014. (Cited on pages 91 and 92.)

Rong Jin, Shijun Wang, and Yang Zhou. Regularized distance metric learning: Theory and algorithm. In *Advances in neural information processing systems (NeurIPS)*, pages 862–870, 2009. (Cited on pages 35 and 39.)

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems (NeurIPS)*, pages 3146–3154, 2017. (Cited on pages 70 and 80.)

Philip A. Knight. The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008. (Cited on page 27.)

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NeurIPS)*, pages 1097–1105, 2012. (Cited on pages 23 and 92.)

Brian Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4): 287–364, 2013. (Cited on pages 2, 3, 9, 19, 24, 32, and 49.)

John Langford and John Shawe-Taylor. PAC-Bayes & margins. In *Advances in neural information processing systems (NeurIPS)*, pages 439–446, 2003. (Cited on page 17.)

Jung-Eun Lee, Rong Jin, and Anil K. Jain. Rank-based distance metric learning: An application to image retrieval. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. (Cited on page 25.)

Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research (JMLR)*, 18(1):559–563, 2017. (Cited on page 51.)

Gaël Letarte, Emilie Morvant, and Pascal Germain. Pseudo-bayesian learning with kernel Fourier transform as prior. In *The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 768–776, 2019. (Cited on pages 26, 60, 61, 62, 63, 66, 67, 69, 71, and 79.)

Jingjing Li, Jidong Zhao, and Ke Lu. Joint feature selection and structure preservation for domain adaptation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1697–1703, 2016. (Cited on pages 82 and 83.)

Kun Liu, Jiangrui Han, Haiyong Chen, Haowei Yan, and Peng Yang. Defect detection on EL images based on deep feature optimized by metric learning for imbalanced data. In *2019 25th International Conference on Automation and Computing*, pages 1–5. IEEE, 2019. (Cited on page 34.)

Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39 (2):539–550, 2008. (Cited on page 34.)

Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information sciences*, 250:113–141, 2013. (Cited on page 34.)

Jiwen Lu, Xiuzhuang Zhou, Yap-Pen Tan, Yuanyuan Shang, and Jie Zhou. Neighborhood repulsed metric learning for kinship verification. *IEEE transactions on pattern analysis and machine intelligence*, 36(2):331–345, 2013. (Cited on pages 25 and 37.)

Prasanta Chandra Mahalanobis. On the generalized distance in statistics. In *National Institute of Science of India*, 1936. (Cited on page 24.)

Llew Mason, Jonathan Baxter, Peter L. Bartlett, Marcus Frean, et al. Functional gradient techniques for combining hypotheses. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 221–246, 1999. (Cited on page 61.)

David A. McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37(3):355–363, 1999. (Cited on pages 15, 16, and 17.)

Brian McFee and Gert R. Lanckriet. Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 775–782, 2010. (Cited on page 34.)

James Mercer. Functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458):415–446, 1909. (Cited on page 21.)

Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781. (Cited on page 26.)

Emilie Niaf, Olivier Rouvière, Florence Mège-Lechevallier, Flavie Bratan, and Carole Lartizien. Computer-aided diagnosis of prostate cancer in the peripheral zone using multiparametric MRI. *Physics in Medicine & Biology*, 57(12):3833–3851, 2012. (Cited on pages 91 and 100.)

Dino Oglic and Thomas Gärtner. Greedy feature construction. In *Advances in neural information processing systems (NeurIPS)*, pages 3945–3953, 2016. (Cited on pages 60 and 70.)

Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010. (Cited on page 95.)

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011. (Cited on page 76.)

Claudio Persello and Lorenzo Bruzzone. Kernel-based domain-invariant feature selection in hyperspectral images for transfer learning. *IEEE transactions on geoscience and remote sensing*, 54(5):2615–2626, 2015. (Cited on pages 82 and 84.)

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems (NeurIPS)*, pages 1177–1184, 2008. (Cited on pages 3, 9, 24, 25, 60, 62, 67, and 79.)

Ievgen Redko, Amaury Habrard, and Marc Sebban. Theoretical analysis of domain adaptation with optimal transport. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 737–753. Springer, 2017. (Cited on page 86.)

Ievgen Redko, Emilie Morvant, Amaury Habrard, Marc Sebban, and Younès Bennani. *Advances in Domain Adaptation Theory.* Elsevier, 2019. (Cited on pages 2 and 82.)

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. (Cited on page 22.)

Walter Rudin. *Fourier analysis on groups*, volume 121967. Wiley Online Library, 1962. (Cited on page 25.)

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision (ECCV)*, pages 213–226. Springer, 2010. (Cited on page 91.)

Robert E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990. (Cited on page 23.)

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999. (Cited on pages 3, 23, 24, and 29.)

Bernhard Schölkopf, Chris Burgest, and Vladimir N. Vapnik. Extracting support data for a given task. In *Proceedings, First International Conference on Knowledge Discovery & Data Mining. AAAI Press, Menlo Park, CA*, pages 252–257, 1995. (Cited on page 49.)

Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *Advances in neural information processing systems (NeurIPS)*, pages 41–48, 2004. (Cited on page 25.)

John Shawe-Taylor and Robert C. Williamson. A PAC analysis of a bayesian estimator. In *Proceedings of the tenth annual conference on Computational learning theory (COLT)*, pages 2–9, 1997. (Cited on page 17.)

Aman Sinha and John C. Duchi. Learning kernels with random features. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1298–1306, 2016. (Cited on pages 26, 60, and 67.)

Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2058–2065. AAAI Press, 2016. (Cited on page 95.)

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1–9, 2015. (Cited on page 92.)

Michel Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Publications Mathématiques de l'Institut des Hautes Etudes Scientifiques*, 81(1):73–205, 1995. (Cited on page 86.)

Selen Uguroglu and Jaime Carbonell. Feature selection for transfer learning. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 430–442. Springer, 2011. (Cited on pages 82, 84, and 87.)

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. (Cited on page 15.)

Cornelis J. Van Rijsbergen. Further experiments with hierarchic clustering in document retrieval. *Information Storage and Retrieval*, 10(1):1–14, 1974. (Cited on pages 11 and 34.)

Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer, 1995. (Cited on pages 13 and 49.)

Vladimir N. Vapnik and Alexey Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2): 264–280, 1971. (Cited on pages 15, 17, and 40.)

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008. (Cited on pages 3, 9, 24, and 26.)

Pascal Vincent and Yoshua Bengio. Kernel matching pursuit. *Machine learning*, 48(1-3):165–187, 2002. (Cited on page 60.)

Robin Vogel, Aurélien Bellet, and Stéphan Clémençon. A probabilistic theory of supervised similarity learning for pointwise ROC curve optimization. In *International Conference on Machine Learning (ICML)*, pages 5065–5074, 2018. (Cited on page 34.)

Nan Wang, Xibin Zhao, Yu Jiang, and Yue Gao. Iterative metric learning for imbalance data classification. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2805–2811, 2018. (Cited on pages 34 and 50.)

Yiru Wang, Weihao Gan, Jie Yang, Wei Wu, and Junjie Yan. Dynamic curriculum learning for imbalanced data classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5017–5026, 2019. (Cited on page 34.)

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. (Cited on pages 49 and 50.)

Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research (JMLR)*, 10(Feb):207–244, 2009. (Cited on pages 25, 32, 35, 37, and 50.)

Di Wu, Boyu Wang, Doina Precup, and Benoit Boulet. Boosting based multiple kernel learning and transfer regression for electricity load forecasting. In *European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 39–51. Springer, 2017. (Cited on pages 60, 61, and 70.)

Shiming Xiang, Feiping Nie, and Changshui Zhang. Learning a mahalanobis distance metric for data clustering and classification. *Pattern recognition*, 41(12):3600–3612, 2008. (Cited on pages 25 and 37.)

Eric P. Xing, Michael I. Jordan, Stuart J. Russell, and Andrew Y. Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems (NeurIPS)*, pages 521–528, 2003. (Cited on pages 25, 37, and 49.)

Zhong Yin, Yongxiong Wang, Li Liu, Wei Zhang, and Jianhua Zhang. Cross-subject EEG feature selection for emotion recognition using transfer recursive feature elimination. *Frontiers in neurorobotics*, 11:19, 2017. (Cited on pages 82 and 84.)

Pourya Zadeh, Reshad Hosseini, and Suvrit Sra. Geometric mean metric learning. In *International conference on machine learning (ICML)*, pages 2464–2471, 2016. (Cited on pages 25, 32, 35, 37, 50, and 57.)

Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Third IEEE international conference on data mining (ICDM)*, pages 435–442. IEEE, 2003. (Cited on page 34.)

Wei-Shi Zheng, Shaogang Gong, and Tao Xiang. Person re-identification by probabilistic relative distance comparison. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 649–656, 2011. (Cited on page 25.)

Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997. (Cited on page 50.)

**Abstract** Machine learning consists in the study and design of algorithms that build models able to handle non trivial tasks as well as or better than humans and hopefully at a lesser cost. These models are typically trained from a dataset where each example describes an instance of the same task and is represented by a set of characteristics and an expected outcome or label which we usually want to predict. An element required for the success of any machine learning algorithm is related to the quality of the set of characteristics describing the data, also referred as data representation or features. In supervised learning, the more the features describing the examples are correlated with the label, the more effective the model will be. There exist three main families of features: the "observable", the "handcrafted" and the "latent" features that are usually automatically learned from the training data. The contributions of this thesis fall into the scope of this last category. More precisely, we are interested in the specific setting of learning a discriminative representation when the number of data of interest is limited. A lack of data of interest can be found in different scenarios. First, we tackle the problem of imbalanced learning with a class of interest composed of a few examples by learning a metric that induces a new representation space where the learned models do not favor the majority examples. Second, we propose to handle a scenario with few available examples by learning at the same time a relevant data representation and a model that generalizes well through boosting models using kernels as base learners approximated by random Fourier features. Finally, to address the domain adaptation scenario where the target set contains no label while the source examples are acquired in different conditions, we propose to reduce the discrepancy between the two domains by keeping only the most similar features optimizing the solution of an optimal transport problem between the two domains.

**Résumé** L'apprentissage automatique consiste en l'étude et la conception d'algorithmes qui construisent des modèles capables de traiter des tâches non triviales aussi bien ou mieux que les humains et, si possible, à un moindre coût. Ces modèles sont généralement entraînés à partir d'un ensemble de données où chaque exemple décrit une instance de la même tâche et est représenté par un ensemble de caractéristiques et un résultat ou étiquette que nous voulons généralement prédire. Un élément nécessaire au succès de tout algorithme d'apprentissage automatique est lié à la qualité de l'ensemble de caractéristiques décrivant les données, également appelé représentation des données. Dans l'apprentissage supervisé, plus les caractéristiques décrivant les exemples sont corrélées avec l'étiquette, plus le modèle sera efficace. Il existe trois grandes familles de caractéristiques : les caractéristiques "observables", les caractéristiques "fabriquées à la main" et les caractéristiques "latentes" qui sont généralement apprises automatiquement à partir des données d'entraînement. Les contributions de cette thèse s'inscrivent dans le cadre de cette dernière catégorie. Plus précisément, nous nous intéressons au cadre spécifique de l'apprentissage d'une représentation discriminatoire lorsque le nombre de données d'intérêt est limité. Un manque de données d'intérêt peut être constaté dans différents scénarios. Tout d'abord, nous abordons le problème de l'apprentissage déséquilibré avec une classe d'intérêt composée de peu d'exemples en apprenant une métrique qui induit un nouvel espace de représentation où les modèles appris ne favorisent pas les exemples majoritaires. Deuxièmement, nous proposons de traiter un scénario avec peu d'exemples disponibles en apprenant en même temps une représentation de données pertinente et un modèle qui généralise bien en boostant des modèles basés sur des noyaux et des caractéristiques de Fourier aléatoires. Enfin, pour traiter le scénario d'adaptation de domaine où l'ensemble cible ne contient pas d'étiquette alors que les exemples sources sont acquis dans des conditions différentes, nous proposons de réduire l'écart entre les deux domaines en ne conservant que les caractéristiques les plus similaires qui optimisent la solution d'un problème de transport optimal entre les deux domaines.