Thèse préparée pour l'obtention du grade de
Docteur de l'Université de Lyon
opérée au sein de l'**Université Jean Monnet**

École Doctorale 488, Science Ingénierie Santé
Spécialité de doctorat : **Informatique**

A Unified View of Local Learning: Theory and Algorithms for Enhancing Linear Models

Une Vue Unifiée de l'Apprentissage Local : Théorie et Algorithmes pour l'Amélioration de Modèles Linéaires

par :
Valentina Zantedeschi

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 18 décembre 2018 devant le jury composé de :

Florence D'ALCHE-BUC	Professeure, Télécom ParisTech	Rapporteure
Marianne CLAUSEL	Professeure, Université de Lorraine	Rapporteure
Marc TOMMASI	Professeur, Université de Lille	Examinateur
Pascal GERMAIN	Chargé de Recherche, Université de Lille	Examinateur
Marc SEBBAN	Professeur, Université de Saint-Étienne	Directeur
Rémi EMONET	Maître de Conférences, Université de Saint-Étienne	Co-encadrant

Acknowledgments

I would like to thank all the people that made the accomplishment of this thesis possible.

First of all, my gratitude goes to the thesis's referees, Marianne Clausel and Florence D'Alché-Buc, and the other members of the jury, Pascal Germain and Marc Tommasi, that took the time to evaluate my work and proposed interesting discussions for future work. I am also grateful to my supervisors, Rémi Emonet and Marc Sebban, for the chance they gave me to enter the world of research in Machine Learning, by accepting me in their team and by advising me. A special thank goes to my colleagues Léo Gautheron and Ambrish Rawat, for their valuable remarks on this dissertation, and to all the researchers that I had the pleasure to work with and that contributed to the results of this thesis.

Of course, a PhD is not only the result of hard work: this achievement depended also on the support and love of the people close to me. My deepest gratitude goes to my family that let me make my way and always stood by me. Moreover, I would like to thank the Kung-Fu family and all my friends in Saint-Étienne with whom I shared great experiences, discussions and food. In particular, I am grateful to Adriana, Ayoub and the Cattelain family for making me feel at home. Finally, Guillaume, thank you for your help, understanding and patience.

A mio fratello Davide.

Contents

Introduction

Machine learning is a field of Artificial Intelligence which encompasses algorithms allowing a computer to perform a task without being manually programmed for. In general, the structure and characteristics of a dataset of collected examples are analyzed in order to extrapolate new information, estimate the probability of certain events and make decisions accordingly. A typical machine learning task is the so-called classification: starting from a set of observations (e.g. text corpora) associated to a target result (e.g. the author), it is possible to learn to predict the results for new observations (e.g. the author of anonymous documents). In practice, mathematical models are inferred to represent the relationships between inputs and expected values or, in general, to estimate the unknown mechanism that generates the data.

Usually, the problems are tackled by formulating them mathematically: the data is represented as points in a feature space, where each feature (or dimension) measures a particular attribute of the instances; the desired learning goal is described by a function which guides the model training. The training data is not simply memorized. As we have access to a limited sample of the underlying distribution of the examples, memorizing the training sample would result in poor performance when faced to new situations. Instead, useful information is extracted from the given dataset in order to generalize well on future data, i.e. have good performance on any sample from the unknown distribution.

Examples of applications of machine learning algorithms can be found in various fields, from Computer Vision (for object tracking, face recognition, etc.) to Signal Processing (e.g. speech recognition) and so on, leading to complex systems integrating different machine learning components, such as virtual personal assistants and autonomous vehicles. Machine learning algorithms are particularly helpful and preferable to their analytical counterparts in the following situations:

- The problem is hard to describe exactly, hence no analytical methodology is available to solve it. An example is the task of object detection in pictures. As the objects and the concepts defining them are not easily depicted (in the sense that possible descriptors cannot be translated from natural language to exhaustive logical forms) and visual conditions vary the characteristics of objects, no exact methods exist for these problems even though object detection is an innate skill of human beings.

- It is faster and cheaper to learn a refined solution than to hard-code it, because the latter requires more human expertise and hard-working. In computer graphics, for instance, it might be more practical to learn how to generate diverse procedural textures than to define all the variety manually.

- The problem may vary with time. Consider the case of evasion attacks to spam mail detectors. Attackers constantly adapt to system changes to keep eluding the filter and machine learning solutions offer dynamic ways to account for this concept drift.

There exist several paradigms for solving these complex tasks. This dissertation focuses on enhancing local learning approaches, a family of techniques that infers models by capturing the local characteristics of the space in which the observations are embedded. The founding assumption of these techniques is that the learned model should behave consistently on examples that are close, implying that its results should also change smoothly over the space. The locality can be defined on spatial criteria (e.g. closeness according to a selected metric) or other provided relations, such as the association to the same category of examples or a shared attribute.

Local learning approaches are known to be effective in capturing complex distributions of the data, avoiding to resort to selecting a model specific for the task. However, state of the art techniques suffer from three major drawbacks: they easily memorize the training set, resulting in poor performance on unseen data; their predictions lack of smoothness in particular locations of the space; they scale poorly with the size of the datasets. The contributions of this dissertation investigate the aforementioned pitfalls in two directions: we propose to introduce side information in the problem formulation to enforce smoothness in prediction and attenuate the memorization phenomenon; we provide a new representation for the dataset which takes into account its local specificities and improves scalability.

Context of the thesis This thesis was carried out in the Data Intelligence team of Laboratoire Hubert Curien UMR CNRS 5516, affiliated to Jean Monnet University of Saint-Etienne and University of Lyon, France. Its works were founded by the french National Research Agency through the ANR projects SOLSTICE[1] (ANR-13-BS02-01), which aims at designing new models and tools for dealing with computer vision tasks, and LIVES[2] (ANR-15-CE23-0026-03), which aims at developing well-founded machine learning frameworks for learning with data observed in multiple views.

[1] https://solstice.univ-st-etienne.fr/

[2] https://lives.lif.univ-mrs.fr/

Outline of the dissertation This manuscript is organized in three principal parts and several appendices. For the sake of consistency, the main body of the dissertation covers the contributions related to local learning. Other relevant contributions are summarized below and are fully reported in the appendices.

Part I gives an overview of the scientific context and of the state of the art relevant to the presented works:

- Chapter 1 introduces the Statistical Learning field, with its assumptions and principles, as well as its challenges and practices. More precisely, it details the workflow for learning statistical models capable of describing the observations at hand and performing well on unseen data, along with the generic frameworks for studying the generalization capabilities of a learned model, beyond its empirical performance;
- Chapter 2 presents the principal solutions for empowering linear models, which take advantage of the scalability of linear models while being able to capture complex data distributions. The chapter particularly focuses on local learning approaches, to which the contributions of this dissertation belong, and highlights their intuitions, advantages and pitfalls.

Part II collects the contributions on local learning based on data partitioning:

- Chapter 3 describes a new metric learning method for overcoming the well known issues of local metric learning, namely their tendency to over-fit and their limited applicability. The approach learns for each pair of regions of the input space convex combinations of previously learned local models. Additionally, smoothness in prediction is enforced by a spatial regularization which encourages models of nearby regions to be similar. The goal is to obtain a similarity or distance for comparing any pair of points which is adapted to the regions the points belong to. A theoretical evaluation of the proposed approach is carried out following the framework of algorithmic robustness for deriving generalization bounds. Moreover, the method is compared empirically to state-of-the-art techniques in terms of regression accuracy on two regression tasks. This work led to the **CVPR16** [219] and **CAp16** [216] publications.
- Chapter 4 proposes a novel decentralized technique for collaboratively learning personalized models over a graph of users which is both effective and communication efficient. The learning problem takes the form of a graph-regularized l_1-Adaboost able to build expressive nonlinear models that take into account the singularities of the data of each user but also tend to follow the decisions of neighboring users in the graph. The associated optimization problem jointly

minimizes the overall empirical error while ensuring the smoothness of the users' models with respect to the similarity graph. We propose a decentralized algorithm based on the Frank-Wolfe algorithm, exploiting the intrinsic sparsity of the updates to obtain an efficient collaborative learning procedure with low communication cost. Furthermore, a graph discovery procedure is proposed to estimate the similarity graph between users if unknown. The algorithm is analyzed with respect to its convergence rate, and communication and memory complexities. Finally, a set of experiments is carried out on a toy dataset to show the efficacy of the proposed method both at learning the personalized models and at estimating the communication graph. These contributions were presented at **CAp18** [215] and at **MLPCD18**.

Part III gathers the contributions based on landmark similarities:

- Chapter 5 addresses the pitfalls of kernel methods by introducing a local Support Vector Machines method which clusters the input space, projects the data on landmarks, and jointly learns a linear combination of local models. The approach defines an explicit mapping to a latent space where the problem is linearly separable. Doing so, the approach exhibits better scalability than standard SVMs based on kernel functions. Using the framework of uniform stability, we show that our SVM formulation comes with generalization guarantees on the true risk. The experiments based on the simplest configuration of our model (i.e. landmarks randomly selected, linear projection, linear kernel) show its competitive performance w.r.t. the state of the art and opens the door to new exciting lines of research. This work was presented at **CAp17** [220].

- Chapter 6 expands the method proposed in Chapter 5 to a multi-view classification setting. The goal is to leverage the complementary information of the different views and to linearly scale with the size of the dataset. The similarity estimates w.r.t. a small set of randomly selected landmarks are carried out one view at a time, before learning a linear SVM in this latent space joining all the views. According to the uniform stability framework, the proposed algorithm is robust to slight changes in the training set, which leads to a generalization bound that depends on the number of views and landmarks. The chapter shows how the described approach can be easily adapted to a missing-view scenario by only reconstructing the similarities to the landmarks. The empirical results, both in complete and missing view settings, highlight the superior performance of the proposed method w.r.t. state of the art techniques, in terms of accuracy and execution time. This work was published at **ECML18** [221].

In the Appendices IV, we report two contributions developed during the thesis but unrelated to local learning line of works:

- Appendix A presents a generic framework for learning from weakly labeled data. This field covers different settings such as semi-supervised learning, learning with label proportions, multi-instance learning, noise-tolerant learning, and so on. The appendix introduces a new risk formulation which boils down to a generalization of the standard empirical risk based on surrogate margin-based loss functions. The new risk allows one to express the reliability on the labels in the objective function and can be utilized to derive different kinds of learning algorithms, depending on the knowledge on the labels. These works were published at **NeurIPS16** [217].
- Appendix B introduces a new defense method against adversarial evasion attacks on Deep Neural Networks, based on practical observations. The proposed defense is easy to integrate into models and performs better than state-of-the-art defenses: it is meant to reinforce the structure of a DNN, making its prediction more stable and less likely to be fooled by adversarial samples. The appendix reports an extensive experimental study proving the efficiency of the method against multiple attacks, in comparison with numerous defenses, both in white-box and black-box setups. The work led to a publication at **AISEC17** [222] and to the release of the open-source library ART [145] for studying the adversarial robustness of DNNs.

Finally, we report notions and proofs necessary to the completeness of three chapters:

- Appendix C reports the derivations of tight Lipschitz constants for two families of metrics that we utilize in Chapter 3: Mahalanobis-like distances and bounded-space bilinear forms. The appendix reports the pre-print [218].
- Appendix D reports the derivation of the upper bound of product space curvature needed in Chapter 4.
- Appendix E reports the derivation of the Lagrangian dual problem and some supplementary material related to Chapter 5.

List of Publications

This dissertation is based on the following publications:

International Conferences

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. "Fast and Provably Effective Multi-view Classification with Landmark-based SVM." European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), 2018.

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. "Beta-risk: a new surrogate risk for learning from weakly labeled data." Advances in Neural Information Processing Systems (NeurIPS), 2016.

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. "Metric learning as convex combinations of local models with generalization guarantees." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

National Conferences

Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. "Decentralized Frank-Wolfe Boosting for Collaborative Learning of Personalized Models." Conférence sur l'Apprentissage Automatique (CAp), 2018.

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. "L^3-SVMs: Landmarks-based Linear Local Support Vectors Machines." Conférence sur l'Apprentissage Automatique (CAp), 2017.

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. "Apprentissage de Combinaisons Convexes de Métriques Locales avec Garanties de Généralisation." Conférence sur l'Apprentissage Automatique (CAp), 2016.

International Workshops

Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. "Communication-

Efficient Decentralized Boosting while Discovering the Collaboration Graph." 2nd NeurIPS Workshop on Machine Learning on the Phone and other Consumer Devices (MLPCD 2), 2018.

Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. "Efficient defenses against adversarial attacks." Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISEC), 2017.

Open-Source Software

"Adversarial Robustness Toolbox", Python
https://github.com/IBM/adversarial-robustness-toolbox

"Multiview Landmark-based Support Vector Machines", Python
https://github.com/vzantedeschi/multiviewLSVM

"Decentralized Adaboost", Python
https://github.com/vzantedeschi/Dada

"Locally Linear Landmark-based Support Vector Machines", Python
https://github.com/vzantedeschi/L3SVMs

"Convex Combinations of Local Models", Python
https://gitlab.univ-st-etienne.fr/vzantedeschi/c2ml/

List of Symbols

$\mathbb{R}$	set of real numbers
$\mathbb{R}^d$	set of d-dimensional real-valued vectors
$\mathbb{N}$	set of natural numbers
$\mathbb{R}^-$	set of nonpositive real numbers
$\mathbb{R}^+$	set of nonnegative real numbers
$\mathcal{D}$	unknown sample true distribution
A^T	transposed of matrix A
A_{ij}	component at row i, column j of matrix A
I	identity matrix
$S \sim \mathcal{D}^m$	a sample drawn i.i.d. from $\mathcal{D}^m$
$\mathbb{E}(.)$	expectation of a random variable
$\mathbb{P}(.)$	probability of an event
$x^{(t)}$	value of the variable x at iteration t
x_i	i^{th} component of vector x
$\mathcal{X}$	input vector space
$\mathcal{X}_{\neq 0}$	input space without the null vector
$\mathcal{Y}$	output space
$\mathbb{I}(.)$	indicator function
$\|\cdot\|_p$	l_p-norm in the specified space
$\|\cdot\|$	an arbitrary norm
$(\#.)$	cardinality of a set
$\|\cdot\|_{\mathcal{F}}$	Frobenius norm
$\langle \cdot, \cdot \rangle$	dot product in a given vector space

$\{R_k\}_{k=1}^{K}$	a partition of $\mathcal{X}$ of K clusters
$\mathcal{H}$	a projection space (or latent space) of variable size

Part I

Background

1 Statistical Machine Learning

We start this manuscript by giving an overview of the scientific context of this thesis: the Statistical Learning field encompasses all techniques aiming at inferring a predictive function from the data using statistical tools. The chapter will go through the main notions and practices of learning, from a finite number of observations, statistical models that are capable of performing the same task on future data. We will see that generic algorithms are usually deployed for learning a model suited for the available data. Nevertheless, several decisions have to be made to ensure that the learned models (i) fit the training instances and (ii) generalize to unseen ones. Firstly, the collected data needs to be representative of the task, and, sometimes, needs to be preprocessed in order to filter the information useful for learning or to reduce the complexity of the algorithm. Secondly, a learning algorithm has to be selected: this includes setting the class of functions among which the final model is chosen for the approximation, and setting the optimization technique for searching over the space of possible solutions. Moreover, a metric of performance and an objective function are formulated to guide the learning, by specifying the desired characteristics of the final model. Thanks to these measures, an empirical evaluation of the model is carried out throughout the learning process, both for guiding and monitoring the fitting and for estimating how the obtained model will perform on future data. In order to obtain an estimate of the actual generalization capabilities of the learned model, the empirical study is performed by splitting the training sample into two and by testing the model on the subset which is not used for fitting the model. In a second moment, we will introduce two frameworks for studying the generalization capabilities of the learned model: the Probably Approximately Correct generalization bounds and the Adversarial Robustness. Both approaches rely on considerations independent from the empirical performance of the model at executing the desired task. Overall, they offer practical tools to assess the generalization properties of the learned model and give some insights into why or why-not would a model generalize.

This overview of Statistical Learning is not meant to be exhaustive: many concepts will be only quickly cited, others omitted, while we will dwell on all the elements necessary to the comprehension of the contributions of this thesis.

This chapter is structured as follows: in Section 1.1, we describe the key elements of statistical learning and illustrate how they are commonly combined for learning models with good performance; we, then, present the theoretical and empirical frameworks for assessing the generalization of a model, beyond its performance at test time in Section 1.2.

1.1 Statistical learning generalities

Statistical Learning techniques learn to perform a task from data using statistical tools. While it is commonly called machine learning, there is still much human expertise and supervision put into the process. Data must be collected, analyzed and preprocessed in order to select a good family of models and an appropriate metric of performance for optimization and evaluation. In this section, we introduce the key elements influencing the quality of the learned model, supposing infinite resources and time.

1.1.1 Data

The arguably fundamental component of Machine Learning is data. Unlike a manually programmed procedure, "a ML model is only as good as the data it is fed"[1]. Machine learning models are learned from data, i.e. samples collected in a consistent way, that the learning algorithm will try to fit, i.e. to describe. It is often said that it was not until the era of big data that ML could finally strive. However, having hands on huge amounts of data is not enough to obtain high quality models. As a matter of fact, the learned model is expected to perform well on any point drawn from an unknown underlying distribution of the data that corresponds to the task. For this reason, it is fundamental that the sample S used for training is representative of the data distribution: strongly biased data leads to biased models. The story of the failure of tank detection system [214] is a monitor of how poorly selected data inevitably gives a model incapable of performing the wanted task[2] (it might perform another one). Sometimes, bias in data is inevitable and the learning algorithms should account for it. For instance, considering inclusivity and fairness of the results, the algorithm should not incur in representation disparity and disparity amplification issues [122]. Consider the case of risk assessment[3]: predictions made by an overall high-quality model might be incorrect on minority groups.

The knowledge on the collected dataset influences the choice of approach for learning. There exist several learning paradigms:

[1] Reynold Xin, https://jaxenter.com/apache-spark-machine-learning-interview-143122.html

[2] another example https://www.wired.com/2009/12/hp-notebooks-racist/

[3] www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

Supervised Learning This setting tackles problems that can be formulated as mapping input observations to output labels. The dataset in this setting is composed of annotated examples $S = \{z_i = (x_i, y_i) \in \mathcal{Z}\}_{i=1}^m$, from which the algorithm learns to predict the target output y_i from the given input x_i. When the domain of possible outcomes $\mathcal{Y}$ is discrete, the mapping is called classification, because it corresponds to affecting a class, or category, to each of the observations. A special case is binary classification, where the output space is limited to two outcomes (usually represented as $\{-1, 1\}$ for practical reasons) and the task generally boils down to distinguishing the objects belonging to a category from those which do not. On the contrary, whenever $\mathcal{Y}$ is continuous, the problem becomes predicting an output as close as possible to the target one, task referred to as regression. The most famous regression technique, widely deployed across scientific fields, is the method of (Regularized) Least Squares, which fits a model by finding the parameters that optimize the sum over all training examples of the squared differences between the predicted outputs and the target ones.

Unsupervised Learning In contrast to the previous setting, unsupervised learning comprises problems where no supervision, i.e. no target labels, is provided in the dataset. Hence, this line of research focuses on finding particular patterns in the data, to better analyze it or to subsequently perform other tasks. Clustering techniques belong to this paradigm: their goal is to partition the dataset into groups based on a similarity criterion. For instance, the well-studied algorithm K-means [124] finds K mean points (or centroids) in $\mathcal{X}$ that together minimize the sum of squared distances between any point of the dataset and its closest mean point. After training, data is partitioned into K clusters by affecting a point to the cluster with the closest centroid.

Weakly Labeled Learning This paradigm deals with datasets where weak supervision is provided, in the sense of the availability of labels and their reliability. Because it may be expensive and tricky to assign a reliable and unique label to each training instance, the data at our disposal for the application at hand may be *weakly labeled*. Learning from weak supervision has received important attention over the past few years [91, 118]. This research field includes different settings: only a fraction of the labels are known (Semi-Supervised learning [229]); we can access only the proportions of the classes in predefined groups (Learning with Label Proportions [154] and Multi-Instance Learning [54]); the labels are uncertain or noisy (Noise-Tolerant Learning [3, 141, 153]); different discording labels are given to the same instance by different experts (Multi-Expert Learning [169]). As a consequence of this statement of fact, the data provided in all these situations cannot be fully exploited using supervised techniques, at the risk of drastically reducing the performance of the learned models. Recently, specific techniques have been proposed to deal with each of the previously cited scenarios [118, 141], but also few generic approaches [153, 217].

Reinforcement Learning Reinforcement learning techniques try to mimic the trial-and-error learning mechanism of living beings. More precisely, the system learns, by a trial-and-error search, which actions should be executed in a given situation in order to maximize a reward, which is not immediate but delayed. This paradigm differs from all others because (i) the supervision is not fixed a priori, but comes from experience by exploration-exploitation policies, and (ii) the focus is not on finding hidden structures in the data. We invite the interested reader to refer to [178] for a thorougher presentation of this setting.

In the scope of this thesis, we focus on the supervised learning setting, even though all contributions might be extended to the weakly labeled setting following our works reported in Annex A. Moreover, data is represented by feature vectors and the training samples are supposed to be independently and identically drawn (i.i.d.) from the true distribution $\mathcal{D}$.

1.1.2 Model

Machine Learning approaches consist in learning a mathematical model able to explain the data. Generally, we make the implicit assumption that the underlying distribution of the data can be described by a model, so that, ultimately, learning is finding the best approximation of this true model. In the supervised learning setting, ML models are called predictive models, because they make predictions on new examples based on the information extracted from training data. Predictions usually take the form of scalars or vectors of values.

Definition 1.1 *(Predictive Model) A predictive model is a mathematical model that captures some properties from the data and, consequently, returns a prediction, or decision, for an input point.*

However, a universal approximator, expressing all kind of relationships within the data, has not been conceived until now, so assumptions need to be made before beginning the modeling process. Specifically, a family of hypotheses has to be selected based on prior knowledge or analysis of the data.

Definition 1.2 *(Hypothesis class) A hypothesis class or family $\mathcal{C}$ is the set of candidate models from which the learning algorithm selects the most suitable model for the task.*

The chosen hypothesis family can be finite or infinite, and contain heterogeneous or homogeneous models. Usually, the hypothesis class consists of a set of models sharing a fixed common architecture and a set of variable parameters which characterize them[4]. In this context, learning boils down to selecting the model with the best values for the parameters, i.e. optimizing their values.

Let us examine one of the simplest hypothesis family for binary classification: the set of linear classifiers. An hypothesis from this family bases its decisions on a linear combination of the feature values of the training points:

$$f(x) = \text{sign}(\langle \theta, x \rangle + b)$$

with $\theta \in \mathbb{R}^d$ the vector of weights assigned to each feature and $b \in \mathbb{R}$ the offset. θ and b describe a unique hyperplane in the input space and are the parameters of this family of hypotheses. The hyperplane, or decision boundary, splits the input space into two subspaces, each assigned to one of the two classes, so that the points belonging to a subspace will be all assigned to that class. Figure 1.1 gives an example of linearly separable data, i.e. a dataset that can be separated by a linear classifier making no errors. Notice that there may exist multiple hyperplanes that separate perfectly the training examples of the two classes. Yet, they might not perform equally well on unseen examples, as the amount of training data is limited and some regions of the class manifolds are not represented. In such scenarios, it is preferable to select the model with the largest margin, i.e. the maximal distance with the closest points to the hyperplane, to guarantee the best generalization error.

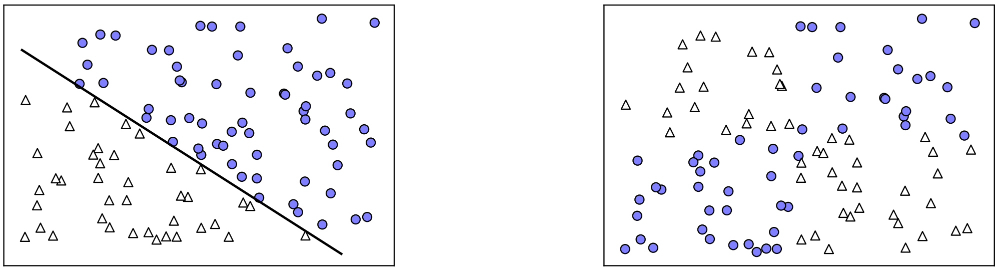


Figure 1.1 – *On the left, linearly separable data can be classified perfectly by at least one linear separator. On the right, a linear classifier is not able to separate the two classes when the class distributions present multi-modalities and non-linearities.*

A linear separator does not intrinsically modify the representation of the data as it can only select and re-weight the features of the input space. Consequently, it performs well only on problems where the data representation is naturally, or by

[4] with the exception of non-parametric models which infer their decisions directly from the training data.

design, appropriate for the task. For instance, the XOR distribution shown in Figure 1.1 cannot be classified correctly by a linear classifier in the original space. It is possible, however, to linearize the problem by applying non-linear transformations on the input space: in this case, by multiplying the two features. There exist more complex models that allow one to learn a separator and a representation suitable to the task at the same time, such as Deep Neural Networks (DNN), which we describe in Annex B, and kernel methods, which we introduce in the next chapter and utilize in the second body of contributions of this manuscript, in Part III.

Choosing the appropriate hypothesis family is, in general, a delicate task, even in autoML works [63, 183], which automatize most of the steps of the learning procedure. Several criteria need to be considered (e.g. its scalability), but, above all, its expressiveness, which determines how well the model will fit the data distribution. A well-studied question on model expressiveness is the adjustment of its complexity, known under the name of bias-variance trade-off: over-simple models (large bias) are unable to capture the data distribution while over-complex ones (large variance) are able to adapt to the noise in the training sample. Both complementary sources of error prevent the learning algorithm from generalizing to unseen data and should be limited by carefully balancing them (see Figure 1.2). In the following, we show how this trade-off can be controlled in the objective function.

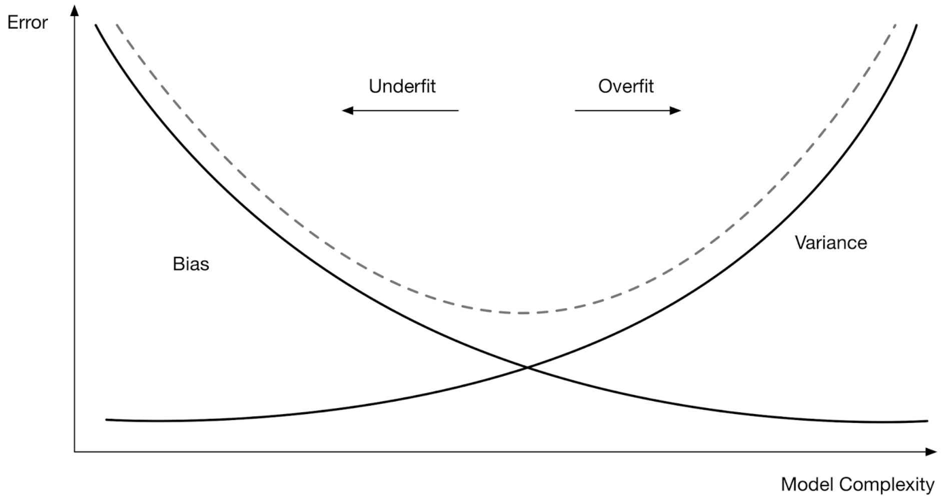


Figure 1.2 – *Intuitive illustration of the bias-variance trade-off. Picture taken from* http://blog.sleptons.com/2016/04/bias-and-variance-in-modeling.html.

1.1.3 Objective Function

Another fundamental element for learning is the formulation of the objective function that shapes the exploration for the best model. Indeed, a metric of performance needs to be chosen to evaluate the quality of any solution f in the hypothesis class $\mathcal{C}$. Because it might be hard to optimize the chosen metric directly, a corresponding loss function is minimized instead which penalizes the model's errors on the training examples. This loss function has to be selected carefully in order to guarantee a good approximation of the metric and, at the same time, a competitive speed-up of the problem resolution. In the supervised learning setting, a loss function assesses the agreement between predicted and target values, by affecting high costs to inadequate predictions and low costs to good ones.

Definition 1.3 *(Loss function) A loss or cost function $\ell : \mathcal{C} \times \mathcal{Z} \to \mathbb{R}^+$ maps any solution $f \in \mathcal{C}$, applied to point $z \in \mathcal{Z}$, to a nonnegative real number.*

Consider, for instance, the case of binary classification. Ideally, the best classifier would be the one scoring the minimal classification error, for which we count one error for every misclassified example $I(f(z) \neq y)$. However, the so-called 0-1 loss is not convex, not differentiable and has zero gradient, which makes it impracticable for optimization. Therefore, other loss functions, called surrogate losses, which are convex and smooth relaxations of the 0-1 loss, are commonly employed instead. Most of them are margin-based (see Figure 1.3), i.e. they penalize the disagreement between target and predicted values measured by the sample margin $yf(z)$. Few examples are the logistic loss (e.g., for the logistic regression [46]), the exponential loss (e.g., for boosting techniques [69]) and the hinge loss (e.g., for the SVM [49]).

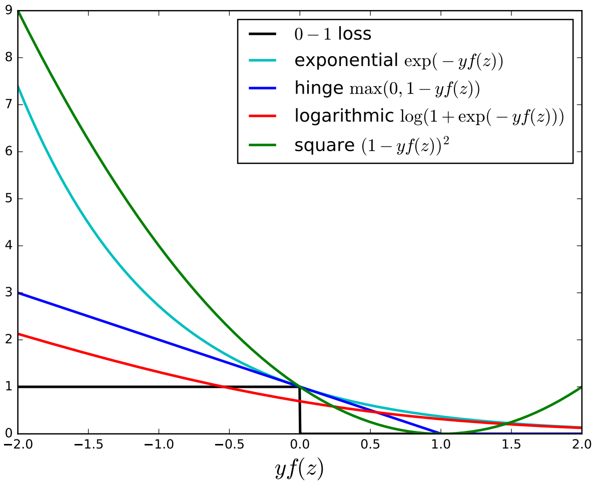


Figure 1.3 – *Illustration of the shapes of common surrogate margin-based losses used in binary classification.*

For obvious reasons, one must carefully choose the metric of evaluation and the loss function, depending on the task at hand. For instance, accounting exclusively for the accuracy rate, and consequently minimizing the error rate, might result in constantly predicting one label in a class-imbalanced setting, where one class is predominant [48].

Once the loss is set, a common approach for choosing the optimal hypothesis f^* from a hypothesis class $\mathcal{C}$ is to select the classifier that minimizes the expected risk over the underlying distribution of the data $\mathcal{D}$ defined over the joint space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.

Definition 1.4 *(True risk) Given a loss ℓ and a data distribution $\mathcal{D}$, the true risk of any hypothesis $f \in \mathcal{C}$ is defined as follows:*

$$R_{\mathcal{D}}(f) = \mathbb{E}_{z \sim \mathcal{D}} \ell(f, z).$$

In practice, as the true distribution is unknown, the true risk cannot be evaluated exactly and an estimate, using the finite training sample S, is optimized instead. This setting goes under the name of Empirical Risk Minimization (ERM).

Definition 1.5 *(Empirical risk) Given a loss ℓ and a sample $S = \{z_i\}_{i=1}^m$ drawn identically and independently (i.i.d.) from $\mathcal{D}$, the empirical risk of any hypothesis $f \in \mathcal{C}$ is defined as follows:*

$$\hat{R}_S(f) = \mathbb{E}_{z \sim S} \ell(f, z) = \frac{1}{m} \sum_{i=1}^{m} \ell(f, z_i).$$

Minimizing the empirical risk corresponds to minimizing the true risk if the training sample S has infinite instances. However, as in practice we work on an approximation of the problem, using a finite amount of data, it is fundamental to study how different the found solution ($\arg\min_f \hat{R}_S(f)$) is from the true one ($\arg\min_f R_{\mathcal{D}}(f)$) and ultimately how it performs on new instances from the distribution of data (how it generalizes). As a matter of fact, a hypothesis with good performance on S (even perfect $\hat{R}_S(f) = 0$) might generalize poorly. In Section 1.2 we will illustrate how one can theoretically relate the empirical risk of a model to its true risk.

From a practical perspective, it is necessary to add constraints, hard or soft, called regularization terms, to the problem formulation in order to prevent memorization, i.e. finding a model that only explains the training sample and has poor performance on unseen examples. In order to avoid over-fitting, the phenomenon of having the true risk much larger than the empirical risk, and because it is harder to control directly the complexity of the family of hypotheses, a common practice is to use an l_p-norm on the parameters of the model to limit its complexity. Such a practice follows the Occam's razor principle, by which a model with reduced complexity is less likely to over-fit the data, as it lacks the expressiveness for it. The Regularized Risk Minimization takes the following form:

$$\min_{f \in \mathcal{C}} \hat{R}_S(f) + \lambda \|f\|$$

where $\lambda \in \mathbb{R}^+$ is a hyper-parameter controlling the trade-off between minimizing the empirical risk (expressive solution) and minimizing the complexity of the solution (simple solution).

Table 1.1 – *Examples of l_p-norms for vectors and matrices.*

Name	Formula	Comments
l_1-norm	$\left\|x\right\|_1 = \sum_j \|x_j\|$	sparsity-inducing, convex, non-smooth
l_2-norm	$\|x\|_2 = \sqrt{\sum_j x_j^2}$	convex, smooth
l_p-norm	$\|x\|_p = \left(\sum_j x_j^p\right)^{\frac{1}{p}}$	convex for $p \geq 1$
Frobenius norm	$\|A\|_{\mathcal{F}} = \sqrt{\sum_{i,j} A_i j^2}$	convex, smooth

In Table 1.1, we report a few examples of l_p-norms for vectors and matrices that we will utilize throughout this manuscript, along with their characteristics and consequences of their deployment.

On the other hand, by giving too much weight to the regularization term, we might encounter the opposite and equally undesired situation of under-fitting the data. It is necessary, then, to tune the hyper-parameters of the objective function using a cross-validation scheme, as we will see later in this section.

Finally, other terms can be added to the objective function to inject additional information and prior knowledge on the task (see Chapters 3 and 4 for two examples) in order to improve the learning and avoid undesired results [116].

1.1.4 Optimization

For most problems a closed-form solution is not available or is impracticable (e.g. requiring to compute too large matrix inversions), which means that the space of hypotheses $\mathcal{C}$ needs to be explored. Even for a finite $\mathcal{C}$, checking the objective function value for any hypothesis in the family is not feasible and a smarter exploration needs to be deployed. Several optimization techniques are at our disposal for finding the minimizer of the objective function starting from an initial solution. We present two of the most-used first-order iterative techniques: Gradient Descent (GD) and Frank-Wolfe (FW) algorithms.

Gradient Descent finds the minimum of a function $R(f)$ by iteratively moving the current solution in the opposite direction of the gradient of the function $\nabla_f R$ with a small step $\gamma \in \mathbb{R}^+$ (called learning rate):

$$f_{i+1} = f_i - \gamma \nabla_f R(f_i).$$

The idea is that the current solution should be improved by searching in the steepest direction with a displacement that depends on the local slope of the curve (see Figure 1.4 for an intuitive illustration). The algorithm can be stopped when the improvement, in terms of minimality of the function, is small enough ($R(f_i) - R(f_{i+1}) \leq \epsilon$). There exist many extensions to the basic algorithm, designed to improve its convergence. We invite the reader to refer to [161] for a survey.

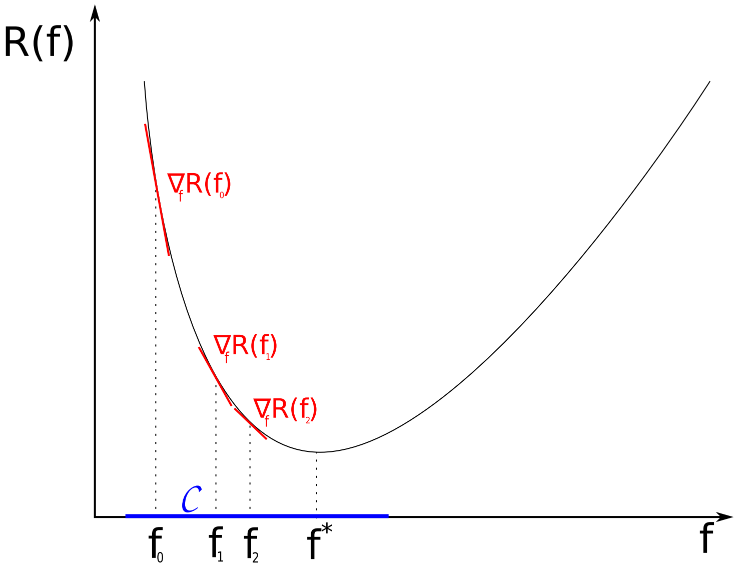


Figure 1.4 – *Illustration of the first iterations of Gradient Descent applied to a mock function $R(f)$.*

The second algorithm is named after the authors who first proposed it in [65]. FW technique iteratively updates the approximate solution by moving it towards the minimizer of the linearized objective function. As shown in Figure 1.5, at each iteration, it selects the minimizer s of the hyper-plane described by the gradient of the function at the current solution, which, by construction, lies on the boundary of the solution domain:

$$s_{i+1} = \arg\min_{f \in \mathcal{C}} \nabla_f R(f_i)^T f$$

$$f_{i+1} = (1 - \gamma) f_i + \gamma s_{i+1}.$$

For convergence, the learning rate $\gamma \in [0, 1]$ should decrease with the number of iterations. Faster variants of FW are available and have been analyzed recently in [105].

The advantages of FW w.r.t. GD are that there is no need for projecting the solution back to the feasible domain (whenever the current solution lies outside it) and that the iterations can be sparse when l_1 constraints are added to the problem.

Notice that the optimization technique affects the performance of the learning process not only in terms of execution time, but also in terms of quality of the model. As a matter of fact, the optimization procedure allows to find an approximation of the exact solution, which can be more or less tight. Reconsider, for instance,

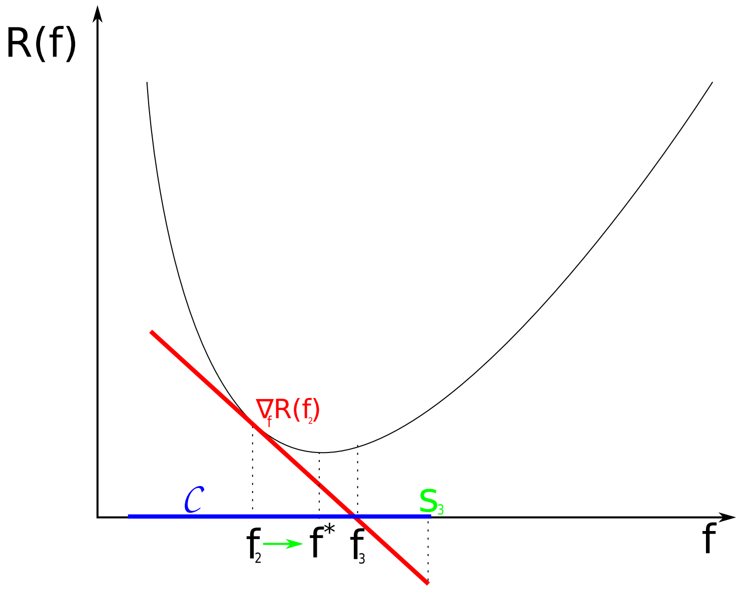


Figure 1.5 – *Illustration of one iteration of Frank-Wolfe applied to the same mock function $R(f)$ as in Figure 1.4 starting from the last approximate solution.*

the role of the learning rate in GD technique: if the learning rate is too large, the optimization might never find the optimum and keep oscillating around it. Moreover, when working with non-convex objective functions and/or non-convex feasible sets, multiple local optima, plateaux and saddle points might exist and the algorithm could get stuck in those regions and never find the global minimum. This last issue can be solved by running the algorithms several times with different initializations and by ensuring that the solution space is explored even after reaching a local minimum.

1.1.5 Overall Learning Pipeline

Once the data is available and the model, objective function and optimization technique have been chosen, a few good practices are necessary to verify that the learned model (i) fits the training data and (ii) performs well on new data. The typical work-flow encompasses the several main steps that we quickly introduce here.

Sampling In order to get an empirical estimation of the generalization capabilities of the learned model, a testing sample must be randomly sampled from the dataset, which is not used for training the model and on which the final model is evaluated. The splitting into training and testing sets can be performed more than once, to get a more accurate evaluation of the performance of the final model. From the training set, we will also sample the so-called validation sets, that are used to check the performance of the algorithm all along the learning process.

Data Preprocessing Raw data might contain examples missing certain attribute values that must be cleaned before learning. This can be done by either removing that attribute from the entire dataset or by imputing the missing values. In this preliminary phase, it might be useful to modify the input space by applying dimensionality reduction techniques, such as Principal Component Analyses (PCA) and Linear Discriminant Analysis (LDA), that help in selecting meaningful features or in creating new ones. Another important step is the rescaling of the different features of the input space to balance their influence in the learning process. Depending on specific needs, one might apply a min-max scaling, in order to rescale each feature to the same range, or standardize each attribute to get zero-mean and one-variance per attribute. It is worth noting that any preprocessing applied to the training sample must invariantly be applied to the testing sample as well.

Hyper-Parameter Tuning Before training the final model, it is necessary to fix the values of the different hyper-parameters of the chosen family of hypotheses and those of the objective function, if there are any. In order to select the most suitable values for the task (those allowing to reach the best test performance), the tuning should be carried out through a cross-validation procedure. Cross-validation consists in repeatedly splitting the training set into another training sample and a validation sample and, for each split, training the model on the new training sample and test it on the validation one. Such procedure is executed for several combinations of values of the hyper-parameters, found using a Grid search, a random search or smarter techniques, such as Gaussian Processes. The combination with the best validation performance, averaged over all the splits, is retained for the final training. Usually k-fold splits are performed, by repeating the training-validation routine k times, with a ratio training sample size over validation sample size equal to $k-1$, and by taking a different $\frac{1}{k}$ fraction of the examples for validation.

Evaluation A systematic evaluation of the learning process and final model must be carried out throughout the learning in order to guide the procedure and to obtain a reliably good model. According to the problem, different metrics of performance should be assessed, such as the accuracy score (or classification error) or F1 measure for classification, the mean squared error for regression, and so on.

1.2 Generalization guarantees

Apart from a rigorous empirical evaluation of a model, it is interesting to assess its generalization capabilities from other perspectives. As a matter of fact, the learning process is carried out on a limited set of examples: it cannot be guaranteed that the final model has good performances on any sample drawn from the underlying data distribution. In this section, we present two general approaches that try to study how well a model generalizes, and somehow try to theoretically explain why that is the case. Both approaches are based on the principle that models are supposed to have similar performances on similar points, thus on similar samples such as the ones drawn from the same distribution.

1.2.1 Generalization Bounds

An hypothesis that achieves low risk on the training sample is not guaranteed to have also low test risk and, in general, to generalize to unseen examples drawn from the true distribution of the data. Over the years, several frameworks have been proposed to provide theoretical guarantees on the generalization capabilities of a learned model. Such guarantees are expressed in the form of upper bounds, called generalization bounds, on the gap between the true risk $R_{\mathcal{D}}$ and the empirical risk $\hat{R}_S$ and depend on several criteria varying from framework to framework. We refer to these bounds as Probably Approximately Correct (PAC) [96, 188] because they stand for a given generalization gap $\varepsilon \in \mathbb{R}^+$ with a certain probability $\delta \in (0, 1]$ (ε and δ being negatively correlated) as follows:

$$\mathbb{P}\left(\left|R_{\mathcal{D}} - \hat{R}_S\right| \geq \varepsilon\right) \leq 1 - \delta.$$

Uniform Convergence The seminal work of Vapnik and Chervonenkis [192, 193] offers a way to derive a generalization bound using a measure of the capacity (or complexity) of the hypothesis class: the VC-dimension. For a binary classification task, the VC-dimension of a hypothesis family is defined as the maximal number of points a model in the family can shatter, i.e. for any labeling of the points the model fits them perfectly, making no errors, and can be estimated for finite or infinite hypothesis classes alike. More formally,

Definition 1.6 *(VC-dimension) The VC-dimension of a hypothesis class $\mathcal{C}$ on an input space $\mathcal{X}$ is given by*

$$VC(\mathcal{C}) = \max\{m | \forall \{x_i \in \mathcal{X}\}_{i=1}^m, \forall y_i \in \{-1, 1\}_{i=1}^m, \exists f \in \mathcal{C} \ s.t. \ f(x_i) = y_i \ \forall i\}.$$

Using this complexity measure and considering the size of the training sample m, a generalization bound can be derived as follows:

Theorem 1.1 *(Uniform Convergence Bound) Given a training sample S of m instances drawn i.i.d. from $\mathcal{D}$ and a hypothesis class $\mathcal{C}$, the following holds true $\forall \delta \in (0, 1]$*

and $\forall f \in \mathcal{C}$ *with probability at least* $1-\delta$

$$R_{\mathcal{D}}(f) \leq \hat{R}_S(f) + \sqrt{\frac{VC(\mathcal{C})\left(ln\frac{2m}{VC(\mathcal{C})}+1\right)+ln\frac{4}{\delta}}{m}}.$$

Intuitively, the more complex the model w.r.t. the amount of available data $\left(\frac{VC(\mathcal{C})}{m}\right)$, the higher the risk of over-fitting the training sample and not generalizing well on unseen instances. In particular for f the minimizer of $\hat{R}_S$ and for a finite $VC(\mathcal{C})$, for the law of large numbers both $R_{\mathcal{D}}(f)$ and $\hat{R}_S$ converge with probability $1-\delta$ to the minimum of $R_{\mathcal{D}}$ over $\mathcal{C}$. Notice that similar but tighter bounds can be formulated using other notions of capacity: the growth function [193], the flat shattering dimension [1], the Rademacher complexity [97], and so on. However, the bounds derived using such complexity measures all suffer from the same flaws. As the capacity is estimated for the entire family of hypotheses (not only for the optimum of the objective function) and the nature of the distribution $\mathcal{D}$ and the quality of the sample S are ignored, the bounds derived using this framework are known to be 'worst-case' bounds, meaning that they are loose upper bounds of the generalization gap $R_{\mathcal{D}} - \hat{R}_S$ evaluated at the selected model.

To improve upon this, more recent frameworks allow to express generalization bounds that are valid exclusively for the minimizer of the considered objective function. As a matter of fact, the formulation of the problem directly impacts the quality of the solution for both training and testing. The two frameworks that we are about to present do not explicitly rely on a complexity term of the chosen hypothesis class but depend on the algorithm deployed for learning the model. A practitioner should make use of the setting that suits the problem the best, considering for instance the regularization terms (e.g. Algorithmic Robustness allows to deal with more complex ones and with l_1 norms).

Uniform Stability We start by presenting the Uniform Stability framework, proposed in [30], that we will utilize in Chapters 5 and 6.

The idea of Uniform Stability is to check if an algorithm produces similar solutions from datasets that are slightly different. If that is the case, the algorithm is less likely to over-fit the training sample and is probably able to generalize to unseen examples from the same distribution. Let S be the original dataset and S^i the set obtained after having replaced the i^{th} example z_i of S by a new sample z_i' drawn from $\mathcal{D}$. We will say that an algorithm is uniformly stable if the difference between the loss suffered (on a new instance) by the hypothesis f learned by the algorithm from S and the loss suffered by the hypothesis f^i learned from S^i converges in $O(\frac{1}{m})$. More formally,

Definition 1.7 *(Uniform Stability) A learning algorithm A has uniform stability of $2\frac{\beta}{m}$ w.r.t. the loss function ℓ with $\beta \in \mathbb{R}^+$ iff*

$$\sup_{z\sim\mathcal{D}} \left|\ell(f,z) - \ell(f^i,z)\right| \leq 2\frac{\beta}{m}.$$

The uniform stability is directly implied if

$$\forall z \in \mathcal{D}, \quad \left|\ell(f,z) - \ell(f^{\setminus i},z)\right| \leq \frac{\beta}{m}$$

where $f^{\setminus i}$ is the hypothesis learned on $S^{\setminus i}$, the set S without the i^{th} instance z_i, which follows from the inequality

$$\left|\ell(f^i,z) - \ell(f,z)\right| \leq \left|\ell(f^i,z) - \ell(f^{\setminus i},z)\right| + \left|\ell(f^{\setminus i},z) - \ell(f,z)\right| \leq 2\frac{\beta}{m}$$

that uses the triangular inequality (at worse, altering a point is like removing a point and adding another one). Using the McDiarmid's concentration inequality [29], it is possible to prove the following theorem:

Theorem 1.2 *[30] Let A be an algorithm with uniform stability $\frac{2\beta}{m}$ w.r.t. a loss ℓ such that $0 \leq \ell(f,z) \leq E$, $\forall z \in \mathcal{Z}$. Then, for any i.i.d. sample S of size m and for any $\delta \in (0,1]$, with probability $1-\delta$*

$$R_{\mathcal{D}}(f) \leq \hat{R}_S(f) + \frac{2\beta}{m} + \Big(4\beta + E\Big)\sqrt{\frac{\ln\frac{1}{\delta}}{2m}},$$

where f is the solution found by A for the sample S.

The convergence of the bound depends on the constant β that accounts for the specificities of the studied problem: the choice of loss function ℓ, the regularization terms and the hyper-parameters. More precisely, it is ultimately determined by the size of the training sample m which should be at least $O(\beta^2)$.

Another well-known framework for studying the generalization guarantees of an algorithm is the Algorithmic Robustness [209] that we use in Chapter 3.

Definition 1.8 *(Algorithmic Robustness) An algorithm A is said to be $(H, \epsilon(.))$-robust, for a convex space $\mathcal{Z}$, $N \in \mathbb{N}$ and $\epsilon : \mathcal{Z}^m \to \mathbb{R}$ iff $\mathcal{Z}$ can be partitioned into H disjoint subsets denoted by $\{R_i\}_{i=1}^H$, such that the following holds for all samples $S \in \mathcal{Z}^m$:*

$$\begin{aligned} &\forall z \in S, \forall z' \in \mathcal{Z}, \forall i = 1, ..., N \\ &if\ z, z' \in R_i\, then\, |\ell(f,z) - \ell(f,z')| \leq \epsilon(S) \end{aligned} \tag{1.1}$$

where f is the solution found by A for the sample S.

In other words, instead of bounding the variation of the solution found for slightly different samples, as in Uniform Stability, in this setting we check if the found solution f has similar predictions on a point z of the training set and a testing point z' close to it in the sense of the partition (see Figure 1.6). Notice that the quantity $\epsilon(S)$ depends on the sample. However, in most derivations, as concentration inequalities are employed, a bound of $\epsilon(S)$ is estimated that accounts for the amount of data m rather than the specificities of the sample.

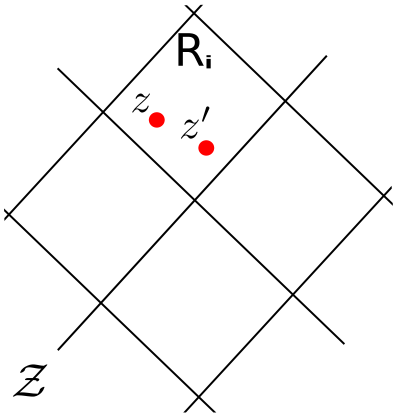


Figure 1.6 – *Given a partition over $\mathcal{Z}$, two points are considered to be similar iff they lie in the same region.*

The following theorem gives the generalization bound based on the notion of Algorithmic Robustness:

Theorem 1.3 *[209] Given a loss function ℓ such that $0 \leq \ell(f,z) \leq E \ \forall z \in \mathcal{Z}$ and $\delta \in (0,1]$, if an algorithm A is $(H, \epsilon(.))$-robust, then with probability $1-\delta$:*

$$R_{\mathcal{D}}(f) \leq \hat{R}_S(f) + \epsilon(S) + E\sqrt{\frac{2Hln2 + 2ln(\frac{1}{\delta})}{m}}$$

where f is the solution of A over S.

Continuity of the used functions Overall, all generalization bounds stand for losses and hypotheses that enjoy some form of 'continuity'. We need to make sure that the chosen functions return similar values when evaluated at similar points, otherwise it is impossible to guarantee consistency in prediction even for samples drawn from the same distribution.

Here we give two properties that we will check for our functions in the contribution-chapters: the Lipschitz continuity of the hypothesis class and the σ-admissibility of the loss function.

Definition 1.9 *(Lipschitz continuity) A function $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$, with $\mathcal{X}$ a convex space, is said to be c-Lipschitz w.r.t. a norm $\|.\|$ if $\exists\, c \in \mathbb{R}, c > 0$ that $\forall x_1, x_2 \in \mathcal{X}$:*

$$|f(x_1) - f(x_2)| \leq c \|x_1 - x_2\| \ .$$

Roughly speaking, a function that is Lipschitz continuous varies slightly within a certain interval.

Considering the σ-admissibility property, given a point, the difference between a loss function evaluated for any two possible hypotheses should be bounded by the difference of hypotheses' predictions, scaled by a constant:

Definition 1.10 *(σ-admissibility) A loss function $\ell(f, z)$ is σ-admissible w.r.t. f if it is convex w.r.t. its first argument and $\forall f_1, f_2$ and if $\forall z \in \mathcal{Z}$:*

$$|\ell(f_1, z) - \ell(f_2, z)| \leq \sigma \|f_1(x) - f_2(x)\| \, .$$

Further thoughts As highlighted throughout this section, the existing theoretical frameworks allow to derive generalization bounds for most of the known learning algorithms. However, they are all independent of the data distribution and the sample at hand, because they all suppose that (i) with enough i.i.d. data instances (tending to infinity) the training sample tends to the real distribution and that (ii) the complexity of the task does not influence the gap between the true and the empirical risk. They fail to describe the behavior of an algorithm when the amount of data is finite, which is the most interesting scenario as it corresponds to the reality. Because of the mentioned flaws, the described theoretical frameworks might lack of significance for certain classes of hypotheses and certain algorithms. For instance, complexity-based settings fail to prove the evident generalization capabilities of Deep Neural Networks [223], giving bounds way too loose to be meaningful. For those models that have almost infinite capacity, recent works [160, 170] suggest that the generalization gap is bounded by the complexity of the task rather than the actual capacity of the model, understood as the potential of fitting any distribution. This hypothesis may find confirmation in the work [170], which highlights a compression-phase in the training of common DNNs.

1.2.2 Adversarial Robustness

Empirically, it is possible to study the generalization properties of a model by inspecting its sensitivity to adversarial examples at testing. Adversarial examples are inputs slightly different from real ones that are maliciously crafted to make a model have a wrong prediction (often, misclassification). Usually, a carefully crafted small perturbation, computed through various techniques [38, 77, 139], is added to the original input (see Figure 1.7).

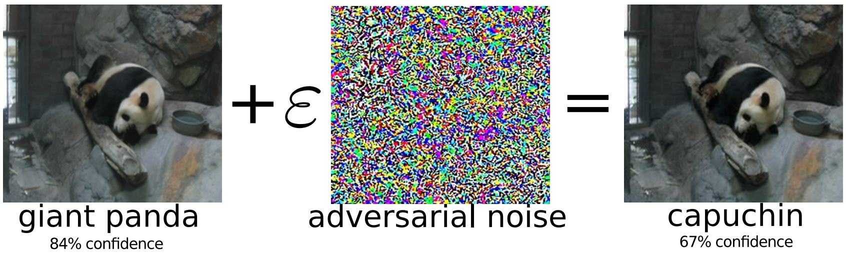


Figure 1.7 – *A clean image of a panda from Imagenet dataset [53] receives adversarial noise. The obtained adversarial image still looks like a panda to the human eye, but will make the model changes its prediction to a relatively confident 'capuchin' label.*

The applied perturbations are, in general, constrained to lie in a p-norm ball with fixed radius as follows:

$$\min_{\|\Delta x\| \leq r} f(x + \Delta x) \neq f(x).$$

As such, the perturbed instances are only slightly different from the original ones (making them hard to detect) and should be classified consistently by the model. Yet, it has been shown that all machine learning methods, although at different scale, are affected by adversarial examples, even those with strong generalization guarantees [23]. If the existence of such test inputs raises security concerns about the deployment of machine learning in real systems, it also stresses the need of a thorougher understanding of artificial learning mechanisms and their actual generalization capabilities.

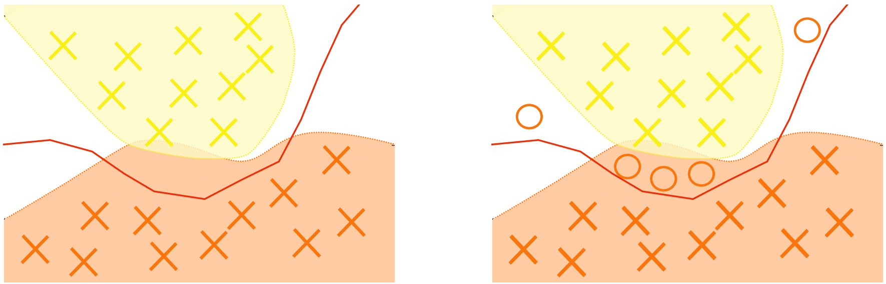


Figure 1.8 – *Illustration of fictitious data distribution and model decision boundary. The colors correspond to the classes and the training instances are marked by crosses. The red line is the decision boundary of the model. Adversarial examples (marked by circles) occur in the regions where the model's decision function and the true distribution of the data do not agree or in regions outside the class manifolds. A possible attacker would try to modify an example so that it lies on the wrong side of the decision boundary.*

As the cause of this phenomenon is still quite obscure, current researches are focused on making models generalize to adversarial examples by making them robust to these malicious perturbations. This body of work is known under the name of Adversarial Robustness.

There exist several metrics for assessing the robustness of a model under several settings (types of attack). The empirical robustness [139] corresponds to the average minimal perturbation that the attacker must introduce to elude the model. The loss sensitivity [101] measures the average loss gradient evaluated on clean examples. Finally, the Cross Lipschitz Extreme Value for nEtwork Robustness metric (CLEVER) [205] estimates a lower bound for the minimal perturbations, based on the extreme value theory. Ideally, a model should yield high values on all these metrics.

Moreover, different techniques, usually called defenses because of their security implications, have been proposed to increase the robustness of a model, such as adversarial training [179], data augmentation in general [136, 222] and robust optimization [128], which train the model to correctly classify also perturbed examples, and architecture modifications [210, 222], which contain the cumulative amplification of errors. However, robustness comes with a worrying price: it has been observed [187] that the models with increased robustness lose on accuracy probably because the defenses act as regularization, making models equally insensitive to all perturbations, ultimately losing their discriminatory capabilities.

1.3 Conclusion

In this chapter, we overviewed the field of Statistical learning, with a particular focus on the Supervised setting. We introduced all significant elements and principles that enable learning, along with the tools for assessing generalization. In particular, we introduced the family of linear models, to which the next chapter is dedicated. We will show how linear models can be empowered using different techniques, without losing their scalability properties. Among such techniques, we will describe local learning approaches, that exploit the local characteristics of the space to improve the expressiveness of the learned model. Specifically, we will give an overview of this setting by presenting its main idea, its challenges and implications before presenting our four contributions that are based on it.

2 Empowering Linear Models

In the previous chapter, we have briefly discussed of the tie between data distribution and model expressiveness. Data characteristics usually vary over the space: the overall distribution might be multi-modal and contain non-linearities. The learning algorithm should be able to capture and adapt to these changes in order to have good performance. Even though linear models fail to describe complex distributions, they are renowned for their scalability, at training and at testing, to datasets big in terms of number of examples and of number of features. Several methods have been proposed to take advantage of the scalability and the simplicity of linear hypotheses to build models with great discriminatory capabilities. These methods empower linear models, in the sense that they enhance their expressive power through different techniques. One of the main advantages of such approaches is that they avoid the hard task of selecting a model appropriate for the task.

In this chapter, we present the three main approaches of this line of work. Such approaches either embed the data in a new representation space or train and combine several models on the original space. We do not include a complete review of the literature, which is reported in the other chapters. Instead, we provide an overview of the notions, ideas and main solutions of this class of approaches.

We start by describing Boosting, a meta-algorithm for learning strong models using weak ones. It trains a set of hypotheses, on modified distributions of the training set to ensure their diversity, and combines their predictions, with a weighted majority vote, for better performance. Boosting methods are widely deployed for their ease of application, performance and theoretical foundation.

The second family that we are going to present is the family of kernel methods, that learn linear models on latent spaces induced by a selected kernel function. In such spaces, the inner product between vectors is given by the chosen kernel function. Consequently, the latent space and its mapping are not needed to be explicitly defined. We will see that models can be trained using the matrix of kernel evaluations for all pairs of points of the training set, called the Gram matrix. Kernel methods are known for their theoretical foundation and their superb expressive power. Yet, their applicability is sometimes limited by their computational and storage burdens. There exist several solutions for scaling kernel methods to big datasets. Most of them propose approximations of the Gram matrix (whose manipulation is often a bottleneck), in order to reduce the costs for inverting it and to limit the number of kernel evaluations.

Lastly, we describe the locally-linear learning family, to which the contributions of this manuscript belong. Local learning techniques focus on capturing the local characteristics of the space to build expressive models. The problem is linearized following the local consistency principle, by which predictions should be consistent for close points. We introduce the two principal solutions for obtaining linear approximations of the problem. We either partition the data and learn a linear model per subset of data or construct a latent space through comparisons between the instances of the dataset and a set of previously selected points, spread over the input space. We will show that local learning approaches have better scalability than kernel methods.

The chapter is organized as follows: in Section 2.1, we present the Boosting meta-learning algorithm and, in particular, AdaBoost; in Section 2.2, we describe the general principles of kernel methods, with their theoretical analysis, and we present the Support Vector Machines method; we additionally give a quick overview of the state of the art techniques for approximating kernel matrices; finally, in Section 2.3, we present locally-linear learning techniques, highlighting their advantages and limitations.

2.1 Boosting methods

Boosting methods are widely-employed supervised learning techniques that belong to the ensemble methods family. The characteristic trait of ensemble methods is that they cope with complex data distributions by constructing a set of n models $H = \{h_j : \mathcal{X} \to \mathbb{R}\}_{j=1}^{n}$ and combining their predictions. The idea is that the final model should be more accurate than the individual models of the ensemble. It has been proved that, in order to build an effective ensemble of models H, (i) each model h_j must be more accurate than a random guessing and (ii) the models must be diverse in their predictions. A strong model can, then, be built from weak, or base, hypotheses.

Definition 2.1 *(Weak Hypothesis) A weak hypothesis on a sample S is any hypothesis trained on S that scores an error slightly smaller than a random guessing.*

In the case of binary classification, a weak model should have a classification error $\hat{\epsilon}_S < 0.5$. Linear models can be utilized as base functions. In the contributions of Chapter 4, we will employ decision stumps, which make predictions based on a threshold value for single features.

Unlike heterogeneous ensemble methods [59], boosting approaches generate the set of models from a single learning algorithm but, to foster diversity, train them on training samples $\{S_j\}_{j=1}^n$, which are modified versions of the original sample S and vary from hypothesis to hypothesis. For instance, AdaBoost [67, 68], the arguably most known boosting technique, iteratively re-samples the training set by re-weighting the examples, depending on the predictions of the weak hypotheses learned so far. For a classification task, at iteration $j+1$, the weight of any $(x_i, y_i) \in S$ is evaluated as follows:

$$w_i^{(j+1)} = w_i^{(j)} \frac{\exp(-\alpha_j y_i h_j(x_i))}{Z^{(j+1)}}$$

with h_j representing the hypothesis learned on S_j of sample weights $w^{(j)}$, $w_i^{(0)} = \frac{1}{m}$ and $Z^{(j+1)} \in \mathbb{R}^+$ a normalization constant so that $\sum_i w_i^{(j+1)} = 1$. $\alpha_j \in \mathbb{R}^+$ is a quality weight affected to the learned weak hypothesis h_j which depends on the classification error $\hat{\epsilon}_{S_j}$:

$$\alpha_j = \frac{1}{2} \log \frac{1 - \hat{\epsilon}_{S_j}}{\hat{\epsilon}_{S_j}} \quad \text{and} \quad \hat{\epsilon}_{S_j} = \sum_{i=1}^m \mathbb{I}[y_i \neq h_j(x_i)].$$

Finally, the weighted vote $\operatorname{sign}(F(.)) = \operatorname{sign}\left(\sum_{j=1}^n \alpha_j h_j(.)\right)$ is used as the prediction rule.

It has been shown [46] that AdaBoost corresponds to fixing the set of base functions H a priori and optimizing the exponential loss, or, equivalently [168], for the strictly increasing monotonicity of the log function:

$$\arg\min_{\alpha} \log\left(\sum_{i=1}^m \exp\left(-(A\alpha)_i\right)\right) \tag{2.1}$$

with $A \in \mathbb{R}^{m \times n}$ the matrix whose entry $a_{ij} = y_i h_j(x_i)$ gives the margin achieved by the classifier h_j on (x_i, y_i). Thus, the sample weight vector w can be rewritten as

$$w = \frac{\exp(-A\alpha)}{\sum_{i=1}^m \exp(-A\alpha)_i}.$$

Many theoretical analyses prove the efficacy of AdaBoost in minimizing the training error on the original sample S [68] and its generalization capabilities [68, 162]. As a matter of fact, even if this meta-algorithm is in contradiction with the Occam's razor principle, it still generalizes well because it maximizes the margins of the training examples [162].

Nowadays, the boosting learning process is usually carried out in its Gradient Boosting formulation [70, 130], which was derived after observing that boosting corresponds to a gradient descent optimization of a particular regression problem. From this standpoint, the learning process is interpreted as a functional gradient descent which fits a hypothesis on the residuals $\{y_i - F(x_i)\}_{i=1}^m$ (i.e. minimizing the squared loss $(y - F(x))^2$) in order to improve the current prediction rule $\text{sign}(F(.))$. The success of gradient boosting is due to the fact that it can extend boosting techniques to any regression loss. By working with the absolute loss $|y - F(x)|$, for instance, it is possible to reduce the sensitivity of the algorithm to outliers.

2.2 Kernel methods

This second family of methods allows to capture complex distributions by using a representation of the data, of a potentially infinite dimensionality, that simplifies the task by making it solvable by a linear model. However, this new representation is not explicitly defined or optimized. On the contrary, such methods make use of a kernel function, fixed before learning. The chosen kernel function allows to directly compute the dot product between vectors on the implicit latent space using the original inputs, without mapping them on this new space. This substitution of the computation of the dot product on the latent space with its evaluation on the original space through the kernel is called kernel trick. As a matter of fact, even for a finite dimensional space, computing the kernel is often cheaper than expressing the mapping between the input space and the latent space and applying it to the input data.

As we will elucidate, kernel methods restrain the set of feasible solutions of their optimization problem to a space defined by the chosen kernel. The advantage is that on this space, the estimation methods are linear. Nevertheless, the choice of kernel function is critical for the quality of the final model. In the following, we briefly report the theoretical foundation of kernel methods, by formally presenting the notions necessary to their understanding and showing how the solutions of their optimization problems can be formulated in terms of evaluations of the kernel function on the training examples.

2.2.1 Theoretical Foundation

The latent space, on which the problem is solved, is determined by a pair-wise function k which needs to satisfy certain conditions. We restrict the following presentation of kernel methods to Mercer's kernel functions, whose definition is reported right away. However, generic kernel functions can take values in the complex space. In the remainder of this manuscript, we will refer to Mercer's kernel functions simply as kernels or kernel functions.

Definition 2.2 *(Mercer's Kernel) A kernel function is any pairwise real-valued function $k : \mathcal{X}^2 \to \mathbb{R}$ which satisfies the following conditions $\forall\ (x_1, x_2) \in \mathcal{X}^2$:*

1. *(symmetry) $k(x_1, x_2) = k(x_2, x_1)$;*
2. *(positive semi-definiteness) $\forall\ m \in \mathbb{N}$ and $\forall\ \{\alpha_i \in \mathbb{R}\}_{i=1}^m$:*

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j k(x_1, x_2) \geq 0.$$

Examples of kernel functions are the Linear kernel and the Radial Basis Function, a.k.a. Squared Exponential Kernel. There exist several variations of their formulations, notably with or without certain scaling and centering factors. We present the formulations that we will utilize in the remainder of this manuscript.

Example 2.1 *(Linear kernel) $\forall\ (x_1, x_2) \in \mathcal{X}^2$:*

$$k(x_1, x_2) = \langle x_1, x_2 \rangle.$$

Example 2.2 *(Radial Basis Function RBF) Given $\gamma \in \mathbb{R}^+$, $\forall\ (x_1, x_2) \in \mathcal{X}^2$:*

$$k(x_1, x_2) = \exp\left(-\frac{\|x_1 - x_2\|_2^2}{\gamma} \right).$$

γ is a radius determining the decay of the kernel and its value is usually fixed depending on the task. One key difference between the two previous functions is that the RBF is a stationary kernel, because it depends on the distance between points no matter their absolute positions in the input space. This is not the case for the Linear kernel.

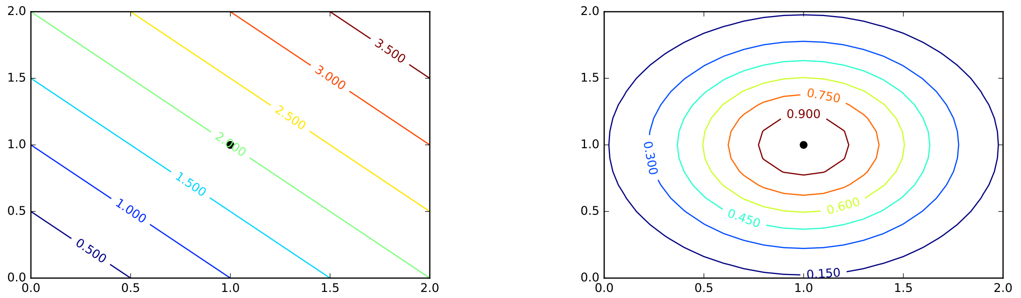


Figure 2.1 – *Illustration of kernel functions evaluated in a 2D space on a fixed point (marked by a dot) and its surrounding points. On the left, the linear kernel induces straight contour lines and its value is unbounded. On the right, the RBF kernel (for $\gamma = 2$) returns its maximal value (1) at the center and has a typical exponential decay as the square distance to the center increases.*

The two conditions in Def. 2.2 are necessary for the validity of Theorem 2.2 that we report in the following. On one hand, they ensure the convexity of the optimization problem and, on the other, the applicability of the so-called kernel trick. They imply the existence of a Hilbert space of potentially infinite dimensionality where the inner product between a pair of projected data points is expressed by the chosen kernel.

Theorem 2.1 *(Mercer's theorem) Any Mercer's kernel $k : \mathcal{X}^2 \rightarrow \mathbb{R}$ is an inner product on a space $\mathcal{H}$ for which there exists a mapping $\mu : \mathcal{X} \rightarrow \mathcal{H}$ such that:*

$$k(x_1, x_2) = \langle \mu(x_1), \mu(x_2) \rangle_{\mathcal{H}}.$$

In general, kernel methods take advantage of the new representation space $\mathcal{H}$ without explicitly expressing the mapping μ. Indeed, common problems can be formulated so that only the inner products between pair of vectors in $\mathcal{H}$ are needed, which can be evaluated directly using $k(.)$.

Here we briefly report the theoretical foundations of kernel methods. In particular, we show how the minimizer of the optimization problem is searched over the Reproducing Kernel Hilbert Space $\mathcal{H}_k$ [5].

Definition 2.3 *(Reproducing Kernel Hilbert Space RKHS) Given a set $\mathcal{X}$, an Hilbert Space $\mathcal{H}_k = \{f : \mathcal{X} \rightarrow \mathbb{R}\}$ of real-valued functions on $\mathcal{X}$ is a Reproducing Kernel Hilbert Space iff $\exists$ k such that $k(x, .) \in \mathcal{H}_k$ and*

$$f(x) = \sum_{i=1}^{\infty} \beta_i k(x, x_i)$$

with $\{\beta_i \in \mathbb{R}\}_{i=1}^{\infty}$.

k is called the reproducing kernel of $\mathcal{H}_k$ and is unique (and vice versa) [165]. Kernel methods restrain the family of hypotheses to the RKHS $\mathcal{H}_k$ corresponding to the chosen kernel. Notice that any function in the RKHS can be expressed by an infinite amount of data from $\mathcal{X}$ which makes the formulation in 2.3 intractable. The following theorem [163] shows how the solution of common optimization problems, which minimize the empirical risk regularized with particular regularizations, depends only on the points of the finite training sample.

Theorem 2.2 *(Representer Theorem) Given a non-empty input set $\mathcal{X}$ and an output set $\mathcal{Y}$, a Mercer's kernel $k : \mathcal{X}^2 \rightarrow \mathbb{R}$ with RKHS $\mathcal{H}_k$ and a training sample $S = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^{m}$, it can be shown that for any $f^* \in \mathcal{H}_k$ minimizing the regularized empirical risk*

$$f^* = \arg\min \hat{R}_S(f) + g(\|f\|_{\mathcal{H}_k})$$

with $g : \mathbb{R}^+ \rightarrow \mathbb{R}$ any monotonically increasing function, $\exists \{\alpha_i \in \mathbb{R}\}_{i=1}^{m}$ so that f^* can be written as

$$f^*(x) = \sum_{i=1}^{m} \alpha_i k(x, x_i).$$

Thus, the solution can be expressed using the training sample and a vector of weights $\alpha = [\ldots, \alpha_i, \ldots] \in \mathbb{R}^m$, each affected to a training instance. The main consequence is that the optimization problem of kernel methods can be reduced to finding the optimal values of the coefficients of α. The problem is, then, linear.

We now give an example of kernel method and show how the vector α can be optimized using the matrix of kernel evaluations $K_{ij} = k(x_i, x_j)$ for all pairs of training points $x_i \in S,\ x_j \in S$. This matrix is often referred to as the Gram matrix of the sample S and is symmetric and positive semi-definite as a direct implication of the Mercer's conditions.

2.2.2 Support Vector Machines

There exist several works exploiting the theory of RKHS. Examples of kernel methods are Gaussian Processes [158], kernel-PCA [164] and RBF network [34]. Among them, Support Vector Machines [28, 49] (SVMs) are well-known for their good performance and theoretical foundation. Here, we present the formulation of SVMs for the case of binary classification, that we will utilize as the basis of two of our contributions and as baseline. However, this technique can be applied to regression and multi-class problems. An interested reader should refer to [165] for a thorougher overview.

SVMs learn linear models that minimize the empirical risk and maximize the margins of the training sample on the representation space (the original input space $\mathcal{X}$ or the latent space $\mathcal{H}$). The idea is that the best separator is the one maximizing the gap between the two classes, i.e. the largest distance with the closest training instances of both classes. As a matter of fact, such a separator should have the best generalization error, as the test instances are more likely to be on the right side of the boundary. The problem can equivalently be formulated as learning two parallel hyperplanes separating the two classes ($\theta^T\mu(x) + b = 1$ for class 1 and $\theta^T\mu(x) + b = -1$ for class -1), with the largest distance between them. This distance is, then, given by $\frac{2}{\|\theta\|}$. Ideally, all points should lie outside the margin created by the two hyperplanes. This not necessarily the case: the data might not be linearly separable in the representation space. This "hard-margin" constraint is relaxed by using a hinge loss on the margin of the examples:

$$\max(0, 1 - y(\theta^T\mu(x) + b)).$$

The soft-margin SVM optimization problem is formulated as follows:

$$\arg\min_{\theta,b} \frac{1}{2}\|\theta\|_2^2 + c\sum_{i=1}^{m} \max(0, 1 - y_i(\theta^T\mu(x_i) + b)) \tag{2.2}$$

where $c \in \mathbb{R}$ is the hyper-parameter controlling the trade-off between margin maximization and empirical risk minimization.

Notice that 2.2 belongs to the set of problems for which Theorem 2.2 holds. In this case, $g(\|f\|_{\mathcal{H}_k}) = \frac{1}{2c}\|\theta\|_2^2)$.

Equivalently, the previous problem is expressed using additional slack variables $\{\xi_i \in \mathbb{R}^+\}_{i=1}^m$, quantifying the error committed by the separator on each training instance (if the margin constraint is satisfied, $\xi_i = 0$):

$$\begin{aligned} &\arg\min_{\theta,b,\xi} \frac{1}{2}\|\theta\|_2^2 + c\sum_{i=1}^{m}\xi_i \\ s.t. \quad & y_i(\theta^T\mu(x_i) + b) \geq 1 - \xi_i \quad \forall i = 1..m \\ & \xi_i \geq 0 \ \forall i = 1..m \end{aligned} \tag{2.3}$$

By solving the previous problem in its dual form, it will become evident how it can be solved without explicitly finding the mapping $\mu : \mathcal{X} \to \mathcal{H}$ and how its solution follows the representer theorem 2.2. The Lagrangian dual problem maximizes the corresponding Lagrangian objective function w.r.t. its Lagrangian multipliers, whose components are equal to 0 when the associated constraints are satisfied and take positive values when they are not. The Lagrangian objective function is expressed as follows:

$$\mathcal{L}(\theta, b, \xi, \alpha, r) = \frac{1}{2}\|\theta\|_2^2 + c\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i(y_i(\theta^T\mu(x_i) + b) + \xi_i - 1) - \sum_{i=1}^{m} r_i\xi_i \tag{2.4}$$

where $\forall i = 1..m$, $\alpha_i \in \mathbb{R}^+$ and $r_i \in \mathbb{R}^+$.

By setting the gradient of $\mathcal{L}$ w.r.t. θ, b and ξ to 0, we find the saddle point corresponding to the function minimum and respecting the constraints:

$$\nabla_\theta\mathcal{L}(\theta, b, \xi, \alpha, r) = \theta - \sum_{i=1}^{m}\alpha_i y_i \mu(x_i)$$

$$\nabla_b\mathcal{L}(\theta, b, \xi, \alpha, r) = -\sum_{i=1}^{m}\alpha_i y_i$$

$$\nabla_{\xi_i}\mathcal{L}(\theta, b, \xi, \alpha, r) = c - \alpha_i - r_i$$

which give

$$\theta = \sum_{i=1}^{m} y_i \alpha_i \mu(x_i) \tag{2.5}$$

$$\sum_{i=1}^{m} y_i \alpha_i = 0 \tag{2.6}$$

$$\alpha_i \leq c \tag{2.7}$$

The dual problem takes the form of a Quadratic Programming problem that can be solved by common optimization techniques. It can be derived by replacing θ by its expression (2.5) and by simplifying with the use of expressions (2.6) and (2.7):

$$\max_{\alpha} \; -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j k(x_i, x_j) + \sum_{i=1}^{m} \alpha_i$$

$$s.t. \quad 0 \leq \alpha_i \leq c \quad \forall i = 1..m$$
$$\sum_{i=1}^{m} \alpha_i y_i = 0 \quad \forall i = 1..m$$

which is concave w.r.t. α and is independent of the explicit mapping $\mu(.)$.

The optimal values of the dual and primal problems are tied due to the following dependency:

$$\max_{\alpha,r} \min_{\theta,b,\xi} \mathcal{L}(\theta, b, \xi, \alpha, r) \leq \min_{\theta,b,\xi} \max_{\alpha,r} \mathcal{L}(\theta, b, \xi, \alpha, r)$$

where the left term corresponds to the optimal value of the dual problem and the right one to the primal. The dual and the primal problems take exactly the same solution at optimality if the Karush-Kuhn-Tucker (KKT) conditions are not violated (see [32]). In the case of SVMs, two additional constraints need to be considered in order to satisfy the KKT conditions:

$$\alpha_i \left(y_i(\theta^T \mu(x_i) + b) - 1 + \xi_i \right) = 0 \quad \forall \; i = 1..m$$

$$r_i \xi_i = 0 \quad \forall \; i = 1..m.$$

Once the Lagrangian dual problem is solved, the characteristic vector θ and the offset b of the optimal margin hyperplane are expressed as follows:

$$\theta = \sum_{i=1}^{m} \alpha_i y_i \mu(x_i)$$

$$b = \theta\mu(x_j) - y_j \quad \text{with} \quad j = \arg\max_{i} \alpha_i.$$

Notice that the solution follows the representer Theorem 2.2. Moreover it depends only on those training instances whose Lagrangian multipliers are non-zero. Such

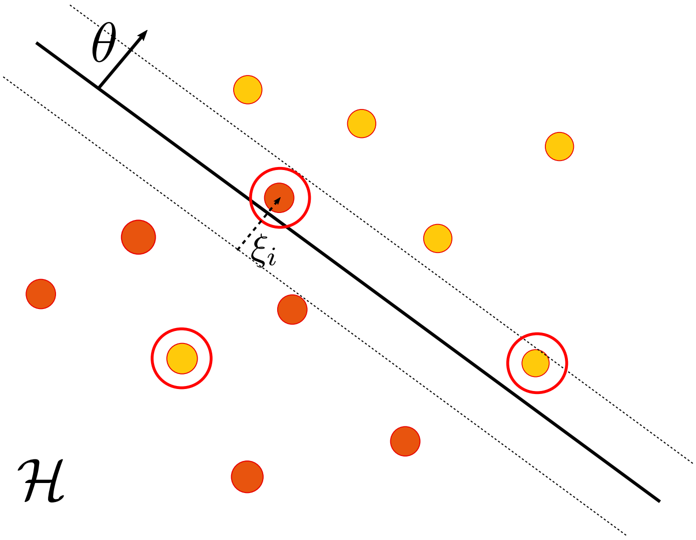


Figure 2.2 – *Possible selected decision hyperplanes and support vectors on a toy problem. The points are represented in the space $\mathcal{H}$ and the colors indicate the two classes. The support vectors (marked by red circles) are all points violating the margin constraint, notably those that are misclassified (they lie on the wrong side of the hyperplane) and those lying within the margin.*

particular training points are called support vectors, from which the method takes its name. Figure 2.2 shows which points could be selected as support vectors in the given non-linearly-separable toy problem.

Finally, the new instances can be classified as

$$f(x) = \operatorname{sign}\left(\langle \theta, \mu(x) \rangle + b\right) = \operatorname{sign}\left(\sum_{i=1}^{m} (\alpha_i y_i k(x_i, x)) + b\right).$$

Notice that we reported SVMs' formulation directly for solving the problem on the projected space $\mathcal{H}$. However, the problem can be solved on $\mathcal{X}$ without mapping the data (i.e. $\mu(x) = x$). In the rest of this manuscript, we will refer to this vanilla formulation as linear SVMs and, whenever the kernel trick is applied, we will use the kernel SVMs designation.

2.2.3 Kernel Approximations

Despite their strong theoretical foundation and effectiveness, kernel methods are known for their lack of scalability to datasets large in number of points (bigger than ca. 10000 instances). From a computational point of view, at training time m^2 evaluations of the kernel need to be carried out to construct the Gram matrix, which needs to be inverted for solving the QP dual problem; at test time, the computational complexity depends on the number of non-zero coefficients of α which are up to m ($O(m)$ for SVMs [174]). From a memory standpoint, the m^2

entries of the Gram Matrix needs to be stored, along with the training instances whose α coefficients are non-zero.

We present, here, the two principal solutions of the literature for scaling kernel methods. They involve low-rank approximations of the Gram matrix, which alleviate the storage and computational burdens at training. Both solutions come with theoretical guarantees on the quality of the approximation and, consequently, the learned model. Notice that the costs for approximating the Gram matrix are not negligible and that, of course, the quality of the approximations depends on the rank of the sketch matrix (higher the rank, better the approximation but higher the complexity of the algorithm and the costs for computing the approximation).

Approximation with Random Fourier features [157] finds a sketch matrix $C \in \mathbb{R}^{2pm}$, such that $K \approx CC^T$, by using a basis set of functions $\{\omega_j \in \mathbb{R}^d\}_{j=1}^p$ drawn i.i.d. from the probability measure induced by the kernel $p(\omega)$. According to Bochner's theorem [125], any point $x \in \mathcal{X}$ admits a Fourier features' representation $z(x) = \frac{1}{\sqrt{p}}[\cos(\omega_1^T x), \ldots, \cos(\omega_p^T x), \sin(\omega_1^T x), \ldots, \sin(\omega_p^T x)]$. This method can be applied only to stationary kernels, in its original formulation, and to dot kernels, of the kind $k(x_1, x_2) = g(\langle x_1, x_2 \rangle)$, in its extensions.

Nyström method [56] can be used to approximate any PSD matrix. It constructs a matrix $C \in \mathbb{R}^{pm}$ by sampling p columns from the Gram matrix, so that $K \approx CO^\dagger C^T$, with $O^\dagger$ the Moore-Penrose pseudo-inverse of the overlap matrix $O \in \mathbb{R}^{p^2}$ between C and C^T in K. Thus, the computational complexity for training the model is sub-quadratic. However, it is unchanged at test times.

2.3 Local Learning

Local learning approaches offer ways to optimize simple models and still capture complex distributions. As already mentioned in this dissertation, they are effective for datasets that present multi-modalities and/or non-linearities because they are able to capture the local characteristics of the space. They are also computationally efficient as they learn only linear classifiers (for which efficient solvers exist).

Local learning methods are based on the local consistency principle: similar points in the feature space should be similar in the output space. This local consistency is typically pursued through two different approaches: one option is to partition the data and learn a model per subset of data; the other option is to extract the local specificities of the space by comparing the instances to a set of points spread over the space. In the following, we thoroughly describe the two approaches.

2.3.1 Local Learning by Data Partitioning

The most classical locally-linear learning family of approaches employs a partitioning of the dataset into subgroups of data and optimize a linear model per subgroup, thereby exclusively exploiting the instances of that group (as shown in Figure 2.3). Such methods embrace the Divide and Conquer principle: with a sufficient amount of subgroups, the set of models has the expressive power to capture the non-linearities and multi-modalities of the space even when optimizing linear models. Indeed, each of the models adapts to a particular subset of data, by capturing its peculiarities. As a consequence, the individual models have good performance only on the subgroup they are learned from. The decision for an instance must be taken depending on the subgroup it belongs to which means that the final model is stationary on each subset individually but not globally.

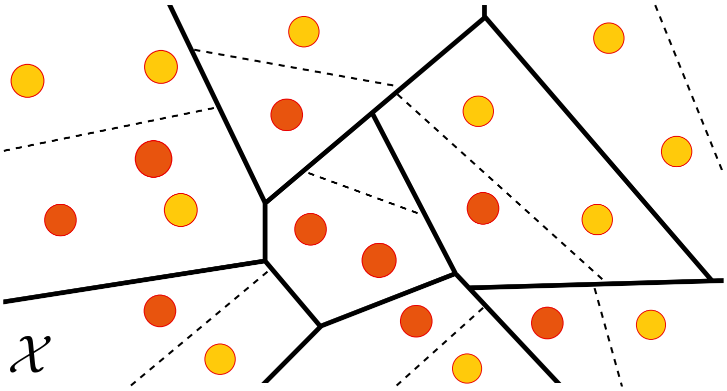


Figure 2.3 – *Example of local and linear classifiers learned on a clustered input space. The points of the toy dataset are represented by circles, with their colors based on their class. The boundaries between regions are drawn as solid lines and the decision boundaries as dashed lines. When the models are learned independently from each other, as in this example, the overall predictions might not be smooth and the models might over-fit the training sample and lack sufficient information for learning.*

A sample can be partitioned following different criteria, based on reasonable observations or prior knowledge about the dataset. For instance, a model could be learned per category of objects in a classification setting, or per task in a multi-task setting (see [39] for an introduction). The splitting could be also carried out using meta-information, such as the id of the user who generated the data, when working with personal data as done in Chapter 4. Arguably the most deployed criterion for splitting the data is the spatial one (as in the contributions of Chapter 3): the instances are grouped based on similarities between feature values, and regions are defined on the input space. Usually, standard clustering techniques, such as K-means, are employed for spatially splitting the input space. Furthermore, the partition can be refined while optimizing the predictive models, as done in [198].

Even though this kind of approaches adapts well to the data distribution, it can present several problems. Firstly, they tend to over-fit, simply because the more the number of models, the less the training data they learn from. Secondly, the overall smoothness in prediction is generally lost, due to decision discrepancies at the boundaries between subsets. Lastly, local models can suffer from other task-specific drawbacks, such as the limitation of comparing points only from the same subset, which can be problematic when learning metrics, as underlined in Chapter 3.

Regularized Locally-Linear Learning A classical solution for overcoming the aforementioned drawbacks is learning a global model jointly with the local models, as done in [41, 80, 156]. By coupling a global with a local model at training and testing, on one hand, an implicit regularization is applied on each local model, by accounting for information coming from outside its subgroup, and, on the other hand, the predictions are smoother over-all the data distribution.

Let us consider, for instance, the optimization problem[1] of Clustered SVMs [80]:

$$\begin{aligned} \underset{\theta,\{\theta_k\}_{k=1}^K}{\arg\min} \; & \frac{\lambda}{2}\|\theta\|_2^2 + \frac{1}{2}\sum_{k=1}^{K}\|\theta_k-\theta\|_2 + c\sum_{k=1}^{K}\sum_{i=1}^{m}\xi_i\;\mathbb{I}[x_i\in R_k] \\ s.t. \quad & y_i\theta_k^T[x_i,1] \geq 1-\xi_i \quad \forall k=1..K \text{ and } \forall x_i\in R_k \\ & \xi_i \geq 0 \;\forall i=1..m \end{aligned} \tag{2.8}$$

with $\lambda \in \mathbb{R}^+$ and $c \in \mathbb{R}^+$ two hyper-parameters.

After having clustered the dataset using K-means, a set of local linear SVMs is jointly optimized with an additional hierarchical regularization constraint: the second term penalizes large differences between an optimized global hyperplane $\theta \in \mathbb{R}^{d+1}$ and the local ones $\{\theta_k \in \mathbb{R}^{d+1}\}_{k=1}^K$. Doing so, the information is bridged between clusters of instances. Notice that, by not optimizing θ and fixing it to 0, the optimization problem (2.8) reduces to learning K independent local SVMs. This approach has proved to be efficient and has good performance. Moreover, it is theoretically founded as it comes with a generalization bound based on the Rademacher complexity. However, forcing all the local models to be equally close to the same vector, with no regard of the commonalties or differences between clusters, can compromise the performance of the local models and cause some undesired behaviors, as shown in the experiments of Chapters 3, 4 and 5.

We argue that softer regularizations lead to better performance. For instance, instead of learning a global model, one could learn combinations of the local ones, based on the topological characteristics of the input space and/or the similarities

[1] The hyperplane offsets are implicit in this formulation

between pairs of subsets. We will discuss further about more effective regularizations in Chapter 3.

2.3.2 Landmark-based Local Learning

Instead of partitioning the data and learning a model per subset of instances, one could optimize a single model capable of extracting the local characteristics and evolving smoothly over the distribution. This can be achieved following, for instance, the similarity principle of kernel methods and metric learning (see Chapter 3 for an introduction): the instances of the dataset are described by comparing them to a set of selected instances, drawn from the true distribution of data or not. In this manuscript, we will refer to such selected points as landmarks, but, in the literature, they are also referred to as anchor points.

Definition 2.4 *(Landmarks) The set of landmarks $\mathcal{L}$ is a set of points $\{l_p \in \mathcal{X}\}_{p=1}^{L}$ called real iff $\forall l_p \in \mathcal{L},\ l_p \sim \mathcal{D}$ or virtual iff they are not drawn from the distribution $\mathcal{D}$.*

Examples of virtual landmarks could be the centroids of regions found through a clustering technique [107] or the principal components computed through PCA [220].

In general, no supervision is provided for the set of landmarks (even when available in the case of real landmarks): their purpose is to produce a representation independent from their labels. Such a representation is built through point-landmark comparisons: the more similar an instance is to a landmark, the more influential that landmark will be in the decision for that instance (as shown in Figure 2.4).

Also this family of approaches takes advantage of the efficiency of linear models. As a matter of fact, the point-landmark comparisons allow for defining a mapping from the original space $\mathcal{X}$ to a latent space $\mathcal{H}$ in which the problem is supposed to be linear. The difference with data partitioning solutions is that the global stationarity is maintained, i.e. the representation of an instance is independent of the subset of data it belongs to. Doing so, the predictions are naturally smooth without the help of regularization constraints. Moreover, the instances are not assigned to a particular local model, unlike with a hard partition, and the predictions can benefit from the information coming from the entire dataset.

In contrast with pure kernel methods, the number of pair-wise comparisons are limited before training the model, because they are performed exclusively between points and landmarks and not between all pairs of training points (Lm comparisons instead of m^2). Furthermore, the predictions are also based on limited comparisons ($O(L)$ instead of $O(m)$). These restrictions imply a better scalability of the approaches w.r.t. the size of the dataset, both at training and test time. Moreover,

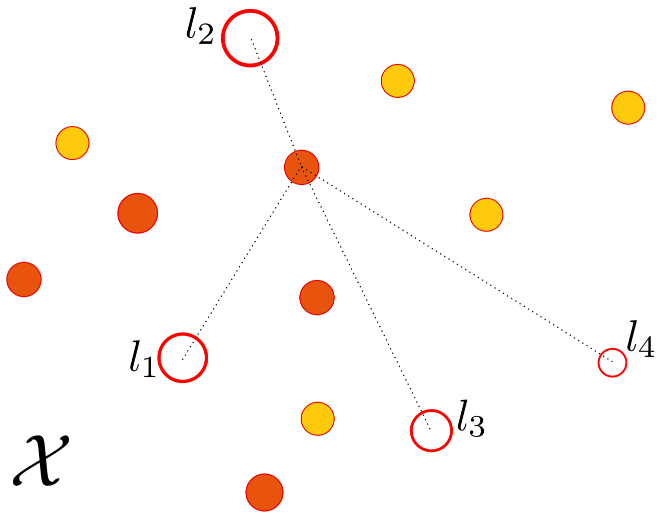


Figure 2.4 – *Illustration of the influence of the selected landmarks in the decisions. The points are represented in the original space $\mathcal{X}$ and the colors indicate the two classes. The landmarks are marked by red circles, whose sizes depend on the similarity with the given point. Each point in the dataset is compared exclusively to the landmarks and its new representation depends on its degree of similarity with each of them.*

they allow for deriving an explicit formulation of the latent space and its mapping, making the study and the interpretation of the learned models easier.

The performance of the landmark-based methods strongly depends on the methods for estimating the relationships between instances and landmarks. We describe, here, the two principal approaches for modeling these relationships.

Local Coordinate Coding Several techniques [107, 199, 213] make use of Local Coordinate Coding (LCC) for modeling the data. A set of landmarks defines a local coordinate system, such that each instance of the dataset can be approximately represented by a linear combination of the landmarks, as follows:

$$\forall x_i \in \mathcal{X},\ LCC(x_i) = \sum_{p=1}^{L} C_{ip} l_p \text{ with } \sum_{p=1}^{L} C_{ip} = 1.$$

The coding is called local because, through different techniques, each instance is expressed using a limited number of landmarks so that the learning is efficient. Generally, a model is, then, optimized per landmark using the training sample encoded with the given LCC. For instance, L linear SVMs can be optimized as

$$
\begin{aligned}
&\operatorname*{arg\,min}_{\{\theta_p, b_p\}_{p=1}^{L}} \frac{1}{2}\sum_{p=1}^{L} \|\theta_p\|_2^2 + c\sum_{i=1}^{m} \xi_i \\
s.t. \quad & y_i \sum_{p=1}^{L} \theta_p^T C_{ip} l_p + b_p \geq 1 - \xi_i \quad \forall i = 1..m \\
& \xi_i \geq 0 \ \forall i = 1..m.
\end{aligned}
\tag{2.9}
$$

The encoding weights are usually computed considering the point-landmark relationships, such as their Euclidean distance or their geodesic distance [190, 199, 228], and their sparsity can be enforced by only considering the neighboring landmarks [107]. However, techniques that learn them jointly with the models [64, 194, 213] have proved to be more effective, even though they create the problem of how estimating the LCC for test points.

$(\epsilon, \gamma, \tau)-$good similarities The point-landmark relationships can be assessed also by means of kernel functions. The resulting explicit mapping from the original space $\mathcal{X}$ to the latent space $\mathcal{H}$ takes the following form, for a given kernel k:

$$\mu_{\mathcal{L}}(.) = [k(., l_1), \dots, k(., l_L)].$$

This mapping is computationally cheaper than the previously described ones (both fixed and optimized coding), but still capable of capturing complex distributions. Moreover, it is straightforward to model test points.

A final advantage of this coding scheme is that it comes with theoretical guarantees. The authors of [9, 10] introduced the $(\epsilon, \gamma, \tau)-$good similarities theoretical framework for studying the ties between the chosen projection and the quality of the linearization of the problem. The "goodness" of the mapping depends on an unknown set of labeled reasonable points P, whose proportion in the underlying distribution $\mathcal{D}$ equals τ. The idea is to check that, according to the chosen kernel, a proportion $1-\epsilon$ of points is in expectation more similar to the reasonable points of the same class than to those of the opposite one. A kernel function (potentially non-PSD) is then said to be $(\epsilon, \gamma, \tau)-$good if it achieves an expected margin of at least γ

$$\mathbb{E}_{(x', y') \in P}(yy'k(x, x')) \geq \gamma$$

with $1-\epsilon$ probability mass of examples drawn from $\mathcal{D}$.

The following theorem guarantees the existence of a linear classifier with true risk arbitrary close to ϵ when a sufficient amount of landmarks is sampled.

Theorem 2.3 *([9]) Let k be a $(\epsilon, \gamma, \tau)-$good kernel on a distribution $\mathcal{D}$ and $\mathcal{L}$ be a set of $L = \frac{2}{\tau}\left(log(2/\delta) + 8\frac{log(2/\delta)}{\gamma^2}\right)$ real landmarks, with $\delta \in (0, 1]$. Then, with* *probability* $1-\delta$*, the embedded data* $\{\mu_{\mathcal{L}}(x)|x \in \mathcal{D}\}$ *admits a linear separator with a margin of at least* $\frac{\gamma}{2}$ *and an error of at most* $\epsilon + \delta$*.*

Notice that such theoretical results are derived for the $0/1$ loss. Extensions of this theory are available, in particular, for the hinge loss [9].

Selecting Landmarks Like in partition-based approaches, two fundamental questions arise when working with a set of landmarks: how many landmarks are sufficient for capturing the variations of the data distribution and how should they be selected. According to [213], the cardinality of $\mathcal{L}$ should at least be equal to the dimensionality of the intrinsic manifold on which the data distribution lies, which is often much smaller than the dimensionality of $\mathcal{X}$. However, it is generally difficult to estimate this intrinsic dimensionality of the data manifold. The theoretical results of [9] (Theorem 2.3) impose that the number of landmarks should be at least $L = \frac{2}{\tau}\left(log(2/\delta) + 8\frac{log(2/\delta)}{\gamma^2}\right)$ to get, with probability of $1-\delta$, a representation on which a linear model has good performance. However, this estimation also depends on the unknown intrinsic complexity of the problem (given here by the constant τ). Moreover, Theorem 2.3 has validity only for real landmarks (points drawn from the input distribution) and not for virtual ones.

2.4 Conclusion

In this chapter, we gave an overview of the main approaches for empowering linear models. Firstly, we described how weak hypotheses can be boosted to form a strong model and highlighted the conditions on the weak hypotheses for obtaining guaranteed generalization. Secondly, we presented the principles and theoretical foundation of kernel methods, and described the widely-deployed SVMs algorithm for learning large-margin linear classifiers. Lastly, we introduced the local learning family of algorithms and discussed how the locality principle can be applied through data partitioning or landmark-based coding.

The contributions of this manuscript consist of new local learning approaches which address the pitfalls of the aforementioned techniques. We chose to work with this family of algorithms because of their intuitive principles, their natural scalability (unlike kernel methods) and their ease of application (the hypotheses are not necessarily weak, unlike in boosting techniques). Moreover, they can be combined with other learning frameworks. As an example, in Chapter 4, we show how local learning and boosting can be coupled to learn local models on personal data.

We focus our works on both families of local learning approaches described in this chapter. Part II is devoted to local learning by data partitioning and studies the effects of learning with smoothing constraints on the model performance. It consists of two principal contributions. In Chapter 3, we propose a novel approach based on convex combinations of local metrics defined on regions of the input space. The learning algorithm accounts for the topological characteristics of the space and fosters smoothness in prediction between nearby regions. In Chapter 4, we introduce a new decentralized algorithm for collaboratively learning personalized models from data naturally partitioned by user. A regularization term, based on a similarity graph between users, promotes smoothness between learned models.

Part III is devoted to scaling SVMs using locally-linear learning by landmark similarity. In Chapter 5, we introduce the formulation of Landmark-based SVMs and analyze its computational and memory complexities, and its discriminatory power, both empirically and theoretically. Additionally, we empirically study the impact on model performance of the amount of landmarks, also depending on the method for selecting them. In Chapter 6, we extend Landmark-based SVMs to the multi-view setting, where data is observed in multiple feature spaces, and propose an imputation technique for adapting our method to the circumstance of missing views, i.e. when views of certain points are missing from the dataset.

Part II

Learning by Partitioning the Space

Learning Convex Combinations of Local Metrics 3

This chapter is based on the publication

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. Metric learning as convex combinations of local models with generalization guarantees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1478–1486, 2016.

In this chapter, we describe a method for overcoming the issues related to data partitioning-based local learning. Through convex combinations of local models regularized considering the topological characteristics of the input space, we enhance typical local learning techniques. Doing so, we obtain smoother predictions, we avoid over-fitting and we improve the discriminatory capabilities of the final models. We focus our work on metric learning, a framework allowing to improve numerous machine learning approaches by optimizing the distances or similarities for the task at hand such that they reflect the specificities of the data. In this field, local metric learning has already been shown to be very effective, especially to take into account non linearities in the data. However, it is well known that local metric learning (i) can entail over-fitting and (ii) face difficulties in comparing two instances that are assigned to two different local models.

Starting from a partition of the space in regions and a model (a score function) for each region, we tackle these issues by defining a metric between points as a weighted combination of the local models. A weight vector is learned for each pair of regions, and a spatial regularization is introduced to ensure that the weight vectors evolve smoothly and that nearby models are favored in the combination. The proposed approach, called Convex Combinations of Local Models (**C2LM**), has the particularities of being defined in a regression setting, of working implicitly at different scales, and of being generic enough to be applicable to both similarities and distances.

The chapter is structured as follows: we first give the background notions of Metric Learning and its state of the art techniques in Section 3.1; in Section 3.2 we illustrate our method and in Section 3.3 prove its theoretical guarantees using the framework of algorithmic robustness; finally, in Section 3.4 we carry out experiments with datasets using both distances (perceptual color distances, using Mahalanobis-like distances) and similarities (semantic word similarities, using bilinear forms), showing that our method consistently improves regression accuracy even in the case when the amount of training data is small.

3.1 A brief introduction to metric learning

In many machine learning tasks, like classification, clustering or ranking, decisions are based on distance or similarity functions, which are indistinctly called *metrics*. These functions are utilized to compare objects, with the aim of determining at which degree the objects should be treated similarly. Intuitively and following the saying "birds of a feather flock together", when an algorithm is faced with examples similar to previously encountered ones, it should identify them and act in a similar way. For instance, one of the simplest non-parametric classifiers, the Nearest Neighbors classifier [50], bases its decisions purely on the classes of the closest labeled inputs, which are selected according to a distance function defined on the input space, such as the Euclidean distance (see Figure 3.1). To classify an unlabeled input, the most frequent class among the points judged relevant is used. It is evident how the choice of metric is critical in the final performance of this classifier.

In general, these fundamental functions can take the form of distances or similarities. We give, in the following, the formal definitions of both notions.

Definition 3.1 *(Similarity) Given a vector space $\mathcal{X} \subset \mathbb{R}^d$, a similarity (or dissimilarity) function is any pairwise real-valued measure $s : \mathcal{X}^2 \to \mathbb{R}$ that quantifies the similarity (or dissimilarity) between any pair $(x_1, x_2) \in \mathcal{X}^2$.*

A similarity measure should take large and positive values for similar objects and large and negative numbers for dissimilar ones. A dissimilarity, conversely, assigns positive values to dissimilar pairs and negative values to similar ones.

Notice that Kernels, presented in 2, are particular types of similarity functions that are symmetric and positive semidefinite.

Definition 3.2 *(Distance) Given a vector space $\mathcal{X} \subset \mathbb{R}^d$, a distance function is any function d, satisfying $\forall x_1, x_2, x_3 \in \mathcal{X}^3$*

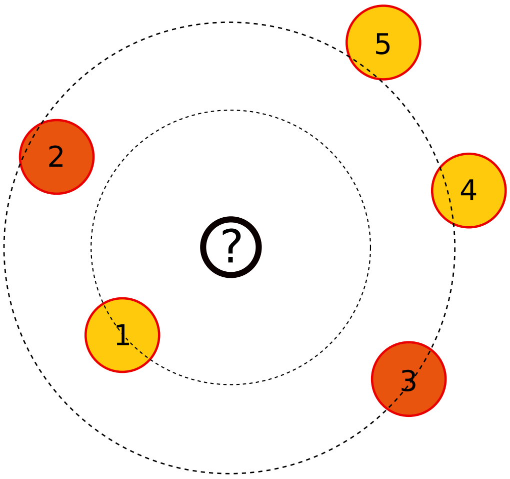


Figure 3.1 – *K Nearest Neighbors classifier: The goal is to classify the instance marked by a question mark. The labeled inputs (the filled circles) are ranked based on their distance from the unlabeled point, computed here as the Euclidean distance. A majority vote of the closest K points is, then, performed to affect a label to the considered point. The choice of the distance in this task is fundamental for meaningful predictions. Moreover, notice how the chosen number of neighbors K influences the final decision.*

1. $d(x_1, x_2) \geq 0$ *(non-negativity);*
2. $d(x_1, x_2) = 0$ *iff* $x_1 = x_2$ *(identity of indiscernibles);*
3. $d(x_1, x_2) = d(x_2, x_1)$ *(symmetry);*
4. $d(x_1, x_2) \leq d(x_1, x_3) + d(x_2, x_3)$ *(triangle inequality).*

Typical examples of distance and similarity functions for feature vectors, i.e. non-structured data, are the Euclidean distance and the Cosine similarity.

Example 3.1 *(Euclidean Distance)*

$$\forall x_1, x_2 \in \mathcal{X}^2, d(x_1, x_2) = \left(\sum_{i=1}^{d} (x_{1i} - x_{2i})^2 \right)^{\frac{1}{2}} = \|x_1 - x_2\|_2$$

expresses the distance between pairs of vectors as the length of the line segment connecting them.

Example 3.2 *(Cosine Similarity)*

$$\forall x_1, x_2 \in \mathcal{X}^2_{\neq 0}, s(x_1, x_2) = \frac{\langle x_1, x_2 \rangle}{\|x_1\|_2 \, \|x_2\|_2}$$

expresses the similarity between non-zero vectors as the cosine of the angle between them, thus it returns values in the interval $[-1, 1]$.

Standard metrics, as the ones just defined, are usually employed because they are easy to set up, as they are non-parametric. Yet, in many cases, these functions fail in reflecting the correspondences between data instances. Moreover, their lack of flexibility makes them unsuitable to some problems. In these cases, it is helpful to learn a parametric metric adapted to the application at hand. In order to enable an algorithm to capture the characteristics of the data, a lot of work has gone during the past ten years into automatically optimizing metric functions, topic referred to as *metric learning* [17, 102]. The idea of such approaches is learning application-dependent distance or similarity functions, that capture the peculiarities of the data and remain robust to data changes, ultimately to make the task easier to solve.

A typical metric learning approach consists of learning a linear Mahalanobis-like metric of the form

$$d_A(x_1, x_2) = \sqrt{(x_1 - x_2)^T A (x_1 - x_2)}$$

parametrized by $A \in \mathbb{R}^{d \times d}$ a positive semi-definite matrix ($A \succeq 0$) whose components are optimized to express the geometry of the space. Notice that for a A that equals to I, the identity matrix, the metric boils down to a Euclidean distance. Furthermore, by applying to A the Cholesky decomposition ($A = L^T L$), it becomes clear how computing this distance function corresponds to computing the Euclidean distance in a new space, where the data has been linearly transformed through L, an upper triangular matrix. It is also possible to enforce additional constraints on the matrix A that result in interesting properties of the distance: for a diagonal A, the distance boils down to weighting the components of the input space and for a low-rank A ($rank(A) < d$), as L becomes rectangular, a dimensionality reduction is applied to the data while computing the distance.

Another common parametric metric is the Bilinear similarity. It takes the form

$$s_A(x_1, x_2) = x_1^T A x_2$$

for a given matrix $A \in \mathbb{R}^{d \times d}$ and can be seen as a generalization of the Cosine similarity.

The $d \times d$ parameters of these metrics can be optimized using auxiliary information: pair-wise constraints (knowing if two instances are similar or dissimilar) or through triplets (considering that from (x_i, x_j, x_l) x_i should be more similar to x_j than to x_l). For instance, the authors of [207] learn a metric that minimizes the distance between similar examples and maximizes the distance between dissimilar ones and show that it improves results in clustering tasks. Other common metric learning frameworks are LMNN (Large-Margin Nearest Neighbors) proposed in [203] for improving K-Nearest Neighbor classification and ITML (Information-Theoric Metric

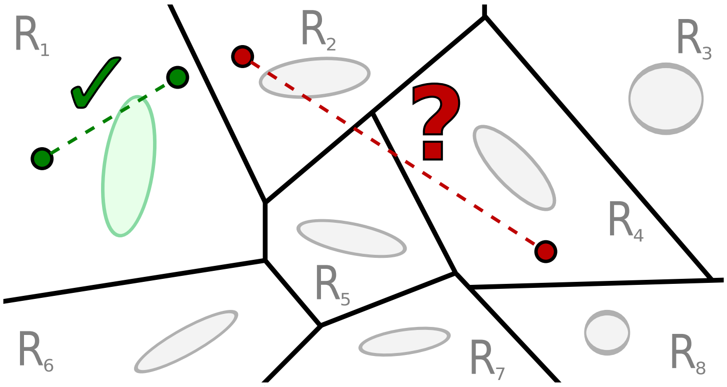


Figure 3.2 – *Limitation of local metric learning: While two points belonging to the same region (e.g. in R_1) can be managed by the corresponding locally-learned metric (depicted as an ellipse), two points from different regions (e.g. in R_2 and R_4) cannot be accurately compared using a single local metric.*

Learning) introduced in [52], which improves KNN and, by means of the LogDet regularization, handles constraints and prior knowledge on the metric.

3.1.1 Local Metric Learning

On most occasions, a unique *global* metric is learned over the input space, typically taking the form of a (linear) geometric transformation. This is also the case also for the previously mentioned LMNN [203] and ITML [52]. However, a global and linear metric, such as the Mahalanobis distance, may not necessarily perform well for all problems, especially when working with data that admits multi-modalities and/or non-linearities. In those cases, *local* metric learning has been shown to be effective because of its flexibility in capturing geometric variations of the input space. Different approaches are viable: learning a metric per instance and use the metric of the closest training input for testing [72, 82]; in a supervised context, learning a metric per class [25, 82]; partitioning the space into regions and learning a metric per region [156, 204].

While these local metric learning approaches can adapt well to variations on the input space, they are quite sensitive to overfitting, especially when the local metrics are learned independently from each other, resulting in non-smooth functions. Some recent solutions have been proposed to alleviate overfitting, for instance by feature space dimensionality reduction [87], by applying manifold regularization [199] or by deploying generative models [147]. However, those approaches mainly focus on improving the results locally, i.e. while comparing instances of the "same region" of the input space. Therefore, they are not well suited to compare points far from each other. This limitation is illustrated in Figure 3.2.

A way to foster smoothness and to obtain globally relevant metrics is learning linear or non-linear combinations of local metrics or kernels (see Multiple Kernel Learning [6] and [84]) for comparing instances. For instance, the authors of [199] propose to jointly optimize a set of basis metrics (one per anchor point) and linear combinations of these metrics (one per input instance) while constraining them to vary smoothly over the instances. Moreover they propose a regularization based on the geometric characteristics of the instance space to ensure smoothness in the decision function. The weight vectors of close instances are then similar and reflect the geometric characteristics of the input space. However, the learned metrics are no longer symmetric and they are only accurate for comparing instances relatively close to each other. Another example is given in [87], where the authors propose to learn linear combinations while controlling the rank of the metric matrices, i.e. the total number of parameters of the problem. Doing so, they penalize too complex solutions, which are probably too tailored for the training instances and thus have lost generalization power on unseen instances.

Both aforementioned frameworks [87, 199] define a linear combination of metrics for each input instance which considerably affects the complexity of their formulations: the number of parameters to be learned increases with the size of the dataset. We claim that the potential accuracy gain is not enough to justify the computational cost and, in any case, it entails some approximations when testing on unseen data (they both assign the weight vector of the closest training instance in term of Euclidean distance).

3.2 Convex combinations of local models

We propose, here, to tackle the previously highlighted issues by learning convex combinations of local metrics that are not only locally good, but also globally relevant. **C2LM** optimizes for any pair of regions a vector of weights corresponding to the contribution of each local model while computing the distance or similarity between two points belonging to those regions (see Figure 3.3). By means of manifold and vector similarity regularization, we constrain the convex combinations to reflect the topological characteristics of the input space and to vary smoothly. Since our principal goal is to learn the influence of each local metric, we will assume in the rest of this chapter that the input space has been previously partitioned into regions and that on each region a local metric has been learned to express its underlying geometry.

Our approach has another peculiarity: unlike the current trend in metric learning, it lies in a regression setting rather than in a classification one. Indeed, it is worth noticing that most metric learning methods use side information brought

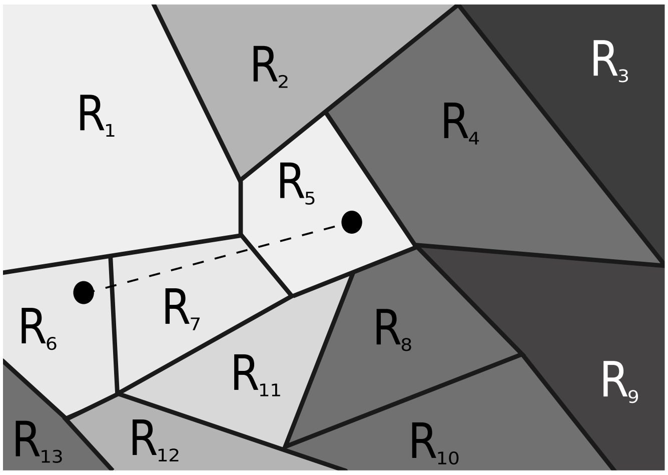


Figure 3.3 – *Illustration of the influence of the local models based on region distances: the more influential a local metric for the learned metric is, the lighter the color of the associated region. For example, the local models of regions R_6, R_5, R_7, R_1 and R_{11} are more influential than those of the other regions, while computing the distance between a point in R_6 and a point in R_5.*

by pairs of training examples in the form of must-link/cannot-link constraints (also called positive/negative pairs) or relative constraints (also called training triplets). A metric learning method typically aims to optimize the parameters of the metric such that it best agrees with those constraints. It turns out that in some applications, the side information provided by the problem of interest simply relies on pairs of examples associated with a target score of (dis)similarity. This is the case in color distance perception (that will constitute one of our two series of experiments), where the training data takes the form of pairs of color patches and their reference perceptual distance ΔE_{00} [167]. This is also the case for databases made of pairs of strings and their corresponding semantic distance (see, e.g., the well known WordSim353 dataset[1]). A last example comes from temporal sequence alignments, where training data can be made of pairs of acoustic signals and their corresponding optimal alignment (e.g. see [109]). In such contexts, state of the art metric learning algorithms face difficulties in accurately capturing the idiosyncrasies of the data. Indeed, the price to pay often implies a dramatic increase in the number of constraints to satisfy. Here, we overcome this issue by dealing with metric learning in a regression setting that allows us to directly fit the target scores.

In the next section, we present our optimization problem for learning convex combinations of local models which takes the form of a least absolute errors regression problem. For the sake of clarity, we first give the notations we will employ in the rest of this chapter.

[1] http://alfonseca.org/eng/research/wordsim353.html

3.2.1 Notations

Let U be the instance-pair space, i.e. the set of pairs $(x_1, x_2) \in \mathcal{X}^2$, and $y : U \rightarrow \mathcal{Y} \subset \mathbb{R}$ be a metric function (the ground-truth metric that can be a distance or a similarity). We assume that U is a compact [66] convex metric space w.r.t. a norm $\|.\|$ and $\mathcal{X} \subset \mathbb{R}^d$. Thus, there exists a constant R such that $\forall x \in \mathcal{X}$, $\|x\| \leq R$. We will refer to $\mathcal{Z} = U \times \mathcal{Y}$ as the set of all possible valued pairs $z = (x_1, x_2, y(x_1, x_2))$, where $(x_1, x_2) \in U$ is a pair of instances and $y(x_1, x_2)$ is the associated target value. We also denote the set of m training pairs by $S = \{z_i \in \mathcal{Z}\}_{i=1}^m$.

3.2.2 Optimization Problem

Let us suppose that the instance space $\mathcal{X}$ has been partitioned into K clusters or regions (for instance using K-Means according to the Euclidean distance), denoted $\{R_k\}_{k=1}^K$ and, on each cluster, a local model $s_k : U \rightarrow \mathbb{R}$ has been defined in order to compare instances belonging to that specific cluster. Let $S = \{s_k(.)\}_{k=1}^K$ be the set of metric functions related to the local models. Our aim is to define for each pair of regions $(R_i, R_j) = R_{ij}$ a metric function $t_{ij} : U \rightarrow \mathbb{R}$ as a convex combination of S and that is also symmetric. The problem we are trying to solve is learning how to compare instances that potentially belong to different clusters. For each pair of regions R_{ij}, we will learn a vector $W_{ij} \in \mathbb{R}^K$ of positive weights representing the contribution of each local model while estimating the similarity between an instance $x_1 \in R_i$ and an instance $x_2 \in R_j$. Therefore, the new metric function $t_{ij}(x_1, x_2)$ related to that pair of regions can be expressed as follows:

$$t_{ij}(x_1, x_2) \; = \; \sum_{k=1}^{K} W_{ijk} s_k(x_1, x_2). \tag{3.1}$$

As we want the overall function to be a metric, we constrain the $K \times K$ matrix of vectors $W = [W_{11}, W_{12}, \ldots, W_{KK}]$ to be symmetric. Thus, $\forall i, j = 1, \ldots, K$, $W_{ij} = W_{ji}$.

We define a loss function $\ell : \mathcal{Z} \rightarrow \mathbb{R}$ over the training set S, corresponding to the gap between t_{ij} and the ground truth metric valued for each pair $z = (x_1 \in R_i, x_2 \in R_j, y(x_1, x_2))$:

$$\begin{aligned} \ell(W, z) &= \ell(W_{ij}, (x_1 \in R_i, x_2 \in R_j, y(x_1, x_2))) \\ &= |t_{ij}(x_1, x_2) - y(x_1, x_2)| \; . \end{aligned} \tag{3.2}$$

Among all possible norms, we choose to define our loss function as an l_1-norm, i.e. the least absolute deviations, because of its robustness to outliers. This loss is assumed to be uniformly upper-bounded by a constant B, i.e. for any pair $z \in \mathcal{Z}$ the deviation of the predicted value from the expected one is finite. We define our optimization problem as follows:

$$\begin{aligned} \arg\min_{W} F_S(W) &= \hat{R}_S(W) + \lambda_1 D(W) + \lambda_2 S(W) \\ s.t. \, \forall i, j = 1, ..., K : &\sum_{k=1}^{K} W_{ijk} = 1 \, and \, W_{ij} \geq 0 \end{aligned} \tag{3.3}$$

where

$$\begin{aligned} \hat{R}_S(W) &= \frac{1}{m} \sum_{z \in S} \ell(W, z) = \\ &= \frac{1}{m} \sum_{i=1}^{K} \sum_{j=1}^{i} \sum_{z \in R_{ij}} \left| \sum_{k=1}^{K} W_{ijk} s_k(x_1, x_2) - y(x_1, x_2) \right| \end{aligned} \tag{3.4}$$

is the mean loss over all training pairs, and

$$D(W) = \sum_{i=1}^{K} \sum_{j=1}^{i} \|E_{ij} \odot W_{ij}\|_2^2 \tag{3.5}$$

$$S(W) = \sum_{i=1}^{K} \sum_{j=1}^{i} \sum_{i'=1}^{K} \sum_{j'=1}^{i'} K_{iji'j'} \left\|W_{ij} - W_{i'j'}\right\|_2^2 \tag{3.6}$$

are two regularizers used to avoid overfitting and λ_1 and λ_2 are the corresponding regularization parameters. The notation $\odot$ indicate the entry-wise product.

The first term, $D(W)$ takes into account the prior influence of each local model in the computation of a weight vector. For instance, for a vector W_{ij} related to the pair of regions (R_i, R_j), we penalize a solution that has big weights associated with the local models that should not be influential in computing the associated metric. As a matter of fact, $E_{ij} \in \mathbb{R}^K$ is a vector whose component E_{ijk} represents the prior influence of the metric s_k. The bigger the component E_{ijk} is, the smaller the learned entry W_{ijk}. E_{ijk} can be estimated in different ways. In the experiments, we base this estimation on the topological characteristics of the decomposition of the space U. As we can see in Figure 3.3, a local model defined on a region close to the pair of regions is more influential than one far from it.

The second term, $S(W)$, expresses the correlations between different weight vectors. Through it, we force the space of vectors of weights to be smooth. In other words, we constrain the vectors defined on close pairs of regions to be similar. As for the prior influence, we base the estimation of the similarity between two vectors W_{ij} and $W_{i',j'}$, expressed by the parameter $K_{iji'j'}$, on the geometric characteristics of the instance space U (see Figure 3.4).

In order to evaluate the prior influence of local models and the similarity between vectors of weights, we need to define a distance function between regions.

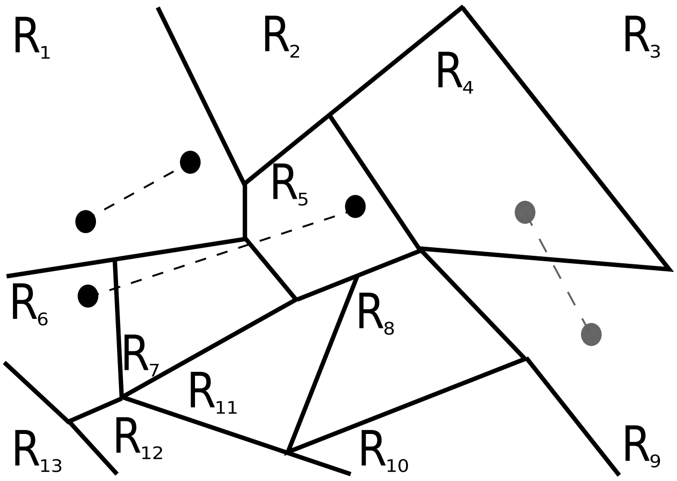


Figure 3.4 – *Similarity of a pair of regions: based on proximity, the vector W_{56} should be more similar to the vector W_{11} (in black) than to the vector W_{49} (in gray).*

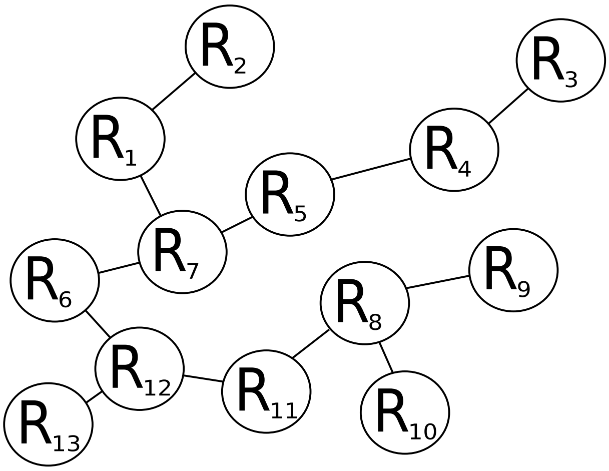


Figure 3.5 – *Minimum Spanning Tree: the distance between two regions corresponds to the number of edges of the shortest path connecting them. E.g., $dist(R_5, R_7) = 1$, $dist(R_{56}, R_4) = dist(R_5, R_4) + dist(R_6, R_4) = 4$ and $dist(R_{56}, R_{49}) = 5$.*

We chose to build the Minimum Spanning Tree of the complete graph of region centroids (computed using the Euclidean distance) and then to express the distance between two regions as the number of edges of the shortest path connecting them (see Figure 3.5). Therefore, for our experiments, we will consider E_{ijk} equal to $dist(R_{ij}, R_k) = dist(R_i, R_k) + dist(R_j, R_k)$ and the similarity $K_{iji'j'} = \exp(-dist(R_{ij}, R'_{i'j}))$ exponentially decreasing with $dist(R_{ij}, R_{i'j'}) = min(dist(R_i, R_{i'}) + dist(R_j, R_{j'}), dist(R_i, R_{j'}) + dist(R_j, R_{i'}))$.

The learned combinations of local models are convex, as we fix their weights to be non-negative and constrain them to sum to one, and the resulting optimization problem 3.3 is convex. Note that the number of parameters to learn depends on the number of regions K defined on the input space ($O(K^3)$). Consequently, the number of constraints is also directly proportional to K^3. This is a main advantage of applying **C2LM** to problems providing pairs of instances and their target score, if we consider the fact that $K \ll m$: in order to adapt the state of the art approaches (meant for classification tasks) to this kind of problems, a number of constraints directly proportional to the number of instances of the dataset has to be added.

3.3 Robustness and generalization bound

In this section, we study the generalization ability of our algorithm according to the notion of Algorithmic Robustness introduced in [209]. As emphasized in Chapter 1, this framework allows us to derive generalization bounds when the variation in the loss associated with two nearby training and testing examples is bounded. This setting is particularly adapted to our framework because it is based on a partition of the input space as we defined it for our problem. The closeness of two examples is based on the notion of covering number. By making use of the Bretagnolle-Huber-Carol inequality and proving that the metric functions $s_k(.)$ are Lipschitz continuous, we can derive a generalization bound for **C2LM**.

3.3.1 Theoretical Guarantees

Let us define a partition of the space $\mathcal{Z}$ of all possible valued pairs $z = (x, x', y(x, x'))$ in order to establish if two pairs of instances are close. The partition is based on the notion of diameter and covering number of a metric space.

Definition 3.3 *(Diameter) The diameter of a metric space $(\mathcal{M}, \rho)$ is defined as:*

$$\text{diam}_\rho(\mathcal{M}) = \sup_{x,x' \in \mathcal{M}} \rho(x - x'). \tag{3.7}$$

Definition 3.4 *(Covering Number [189]) For a metric space $(\mathcal{M}, \rho)$, and $T \subset \mathcal{M}$, we say that $\hat{T} \subset \mathcal{M}$ is a γ-cover of T if $\forall t \in T$, $\exists \hat{t} \subset \hat{T}$ such that $\rho(t, t') \leq \gamma$. The γ-covering number of T is*

$$\mathcal{N}(\gamma, T, \rho) = \min\{\#\hat{T} | \hat{T} \textit{ is a } \gamma\textit{-cover of } T\}. \tag{3.8}$$

In other words, the γ-covering number of a metric space corresponds to the minimal number of regions of radius at most $\gamma > 0$ needed to cover it.

In order to define the closeness between instances of a metric space $\mathcal{Z} = U \times Y$, both the input U and the target Y spaces have to be partitioned. In most works [15, 129, 140, 144], Y is the finite set of labels, so its covering number is exactly equal to $\#Y$ and two instances are considered close if they have the same label. In our setting, we partition the space U into $\mathcal{N}(\gamma_1/2, U, \|.\|_2)$ subsets and the space Y into $\mathcal{N}(\gamma_2/2, Y, |.|)$, so that any region of U (resp. Y) has a diameter smaller than γ_1 (resp. γ_2). In this way, if $z = (x_1, x_2, y(x_1, x_2))$ and $z' = (x'_1, x'_2, y(x'_1, x'_2))$ belong to the same subset of $\mathcal{Z}$, then $\|x_1 - x'_1\|_2 \leq \gamma_1$, $\|x_2 - x'_2\|_2 \leq \gamma_1$ and $|y(x_1, x_2) - y(x'_1, x'_2)| \leq \gamma_2$. In the rest of this chapter, we will refer to $H = \mathcal{N}(\gamma_1/2, U, \|.\|_2)\mathcal{N}(\gamma_2/2, Y, |.|)$ as the covering number of $\mathcal{Z}$.

The following concentration inequality provides a probability bound on the deviation of a multinomial random variable from its expected value. We will use it to obtain information about the theoretical distribution of the valued pairs $z \in \mathcal{Z}$ over the regions of the partition.

Proposition 3.1 *([189]) Let $(|M_1|), ..., |M_H|)$ an i.i.d. multinomial random variable with parameters m and $(p(C_1), ..., p(C_H))$. By the Bretagnolle-Huber-Carol inequality we have: $\mathbb{P}(\sum_{i=1}^{H} \left|\frac{|M_i|}{m} - p(C_i)\right| \geq \lambda) \leq 2^H \exp \frac{-n\lambda^2}{2}$, hence with probability at least $1 - \delta$,*

$$\sum_{i=1}^{H} \left| p(C_i) - \frac{|M_i|}{m} \right| \leq \sqrt{\frac{2H \ln 2 + 2\ln(1/\delta)}{m}}. \tag{3.9}$$

We can now derive a PAC generalization bound for **C2LM**. We first prove that our algorithm is robust, which requires to prove that $\forall k = 1, ..., K : s_k(.)$ is θ_k-Lipschitz (refer to Chapter 1 for a definition). This property is fundamental for the robustness of our algorithm: the fact that the functions $S = \{s_k(.)\}_{k=1}^{K}$ are θ_k-Lipschitz continuous implies that any linear combination of them returns similar values when evaluated on instances belonging to the same region of the partition. According to the nature of the local metric functions $s_k(.)$, the proof of θ_k-Lipschitzness varies. In Sections 3.3.2 and 3.3.3, we will instantiate $s_k(.)$ with Mahalanobis-like distances and bilinear similarities. The derivations of the Lipschitz continuity of such metrics can be found in Appendix C.

Lemma 3.1 *If $\forall k = 1, ..., K$, $s_k(.)$ is θ_k-Lipschitz w.r.t. the norm $\|.\|_2$, the optimization problem (3.3) is $(H, \theta\sqrt{2}\gamma_1 + \gamma_2)$-robust, with $\theta = \max_{k=1..K} \theta_k$.*

Proof. We can partition $\mathcal{Z}$ into $H = \mathcal{N}(\gamma_1/2, U, \|.\|_2)\mathcal{N}(\gamma_2/2, Y, |.|)$ disjoint subsets, such that if $z = (x_1, x_2, y(x_1, x_2))$ and $z' = (x'_1, x'_2, y(x'_1, x'_2))$ belong to the same subset C_h, then $x_1, x'_1 \in R_i$ so $\|x_1 - x'_1\|_2 \leq \gamma_1$, also $x_2, x'_2 \in R_j$ so $\|x_2 - x'_2\|_2 \leq \gamma_1$ and $|y(x_1, x_2) - y(x'_1, x'_2)| \leq \gamma_2$. We have, then:

$$\left|\ell(W_{ij},z)-\ell(W_{ij},z')\right| = \tag{3.10}$$

$$\left|\left|\sum_{k=1}^{K} W_{ijk}s_k(x_1,x_2)-y(x_1,x_2)\right|-\left|\sum_{k=1}^{K} W_{ijk}s_k(x_1',x_2')-y(x_1',x_2')\right|\right|$$

$$\leq \left|\sum_{k=1}^{K} W_{ijk}s_k(x_1,x_2)-\sum_{k=1}^{K} W_{ijk}s_k(x_1',x_2')-y(x_1,x_2)+y(x_1',x_2')\right| \tag{3.11}$$

$$\leq \left|\sum_{k=1}^{K} W_{ijk}\left(s_k(x_1,x_2)-s_k(x_1',x_2')\right)\right|+\left|y(x_1,x_2)-y(x_1',x_2')\right|$$

$$\leq \sum_{k=1}^{K}\left|W_{ijk}\right|\left|s_k(x_1,x_2)-s_k(x_1',x_2')\right|+\gamma_2 \tag{3.12}$$

$$\leq \sum_{k=1}^{K}\left|W_{ijk}\right|\theta_k\left\|\begin{pmatrix}x_1\\x_2\end{pmatrix}-\begin{pmatrix}x_1'\\x_2'\end{pmatrix}\right\|_2+\gamma_2 \tag{3.13}$$

$$\leq \theta\left\|\begin{pmatrix}x_1\\x_2\end{pmatrix}-\begin{pmatrix}x_1'\\x_2'\end{pmatrix}\right\|_2\sum_{k=1}^{K} W_{ijk}+\gamma_2 \tag{3.14}$$

$$\leq \theta\sqrt{2}\gamma_1+\gamma_2\,. \tag{3.15}$$

Eq. (3.11) is due to the reverse triangle inequality. Inequality (3.13) is valid because s_k is multi-variate θ_k-Lipschitz continuous w.r.t. the norm $\|.\|_2$ (see below). In Eq. (3.14), we define $\theta=\max_{k=1,\dots,K}\theta_k$ and recall that $\forall i,j=1,\dots,K: W_{ij}\geq 0$. Eq. (3.15) is due to $\sum_{k=1}^{K}W_{ij}=1$. $\sqrt{2}\gamma_1$ is the maximum $\|.\|_2$ distance between the two vectors. □

We can now derive the generalization bound of **C2LM**.

Lemma 3.2 *As $F_S(W)$ (3.3) is $(H,\theta\sqrt{2}\gamma_1+\gamma_2)$-robust and the training set S is obtained from m i.i.d. draws according to a multinomial random variable, for any $\delta>0$ with probability at least $1-\delta$, we have:*

$$|R_{\mathcal{D}}(W)-\hat{R}_S(W)|\leq\theta\sqrt{2}\gamma_1+\gamma_2+B\sqrt{\frac{2H\ln 2+2\ln 1/\delta}{m}}\,. \tag{3.16}$$

Proof.

$$|R_{\mathcal{D}}(W)-\hat{R}_S(W)| =$$

$$=\left|\mathbb{E}_{z\sim\mathcal{D}}\left(\ell(W,z)\right)-\frac{1}{m}\sum_{z'\in S}\ell(W,z')\right|$$

$$=\left|\sum_{i=1}^{K}\sum_{j=1}^{K}\mathbb{E}\left(\ell(W_{ij},z\in R_{ij})\right)p(R_{ij})-\frac{1}{m}\sum_{i=1}^{K}\sum_{j=1}^{K}\sum_{z'\in R_{ij}}\ell(W_{ij},z')\right|$$

$$
\begin{aligned}
&= \left| \sum_{i=1}^{K} \sum_{j=1}^{K} \mathbb{E}\left(\ell(W_{ij}, z \in R_{ij})]\right) p(R_{ij}) - \sum_{i=1}^{K} \sum_{j=1}^{K} \mathbb{E}\left(\ell(W_{ij}, z \in R_{ij})\right) \frac{m_{ij}}{m} \right| \\
&\quad + \left| \sum_{i=1}^{K} \sum_{j=1}^{K} \mathbb{E}\left(\ell(W_{ij}, z \in R_{ij})\right) \frac{m_{ij}}{m} - \frac{1}{m} \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{z' \in R_{ij}} \ell(W_{ij}, z') \right| \qquad (3.17) \\
&\leq \left| \sum_{i=1}^{K} \sum_{j=1}^{K} \mathbb{E}\left(\ell(W_{ij}, z \in R_{ij})\right) \left(p(R_{ij}) - \frac{m_{ij}}{m} \right) \right| \\
&\quad + \left| \frac{1}{m} \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{z, z' \in R_{ij}} \mathbb{E}\left(\ell(W_{ij}, z) - \ell(W_{ij}, z') \right) \right| \\
&\leq \max_{z \in R_{ij}} \left(\ell(W_{ij}, z)\right) \sum_{i=1}^{K} \sum_{j=1}^{K} \left| p(R_{ij}) - \frac{m_{ij}}{m} \right| \\
&\quad + \frac{1}{m} \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{z, z' \in R_{ij}} \max(\ell(W_{ij}, z) - \ell(W_{ij}, z')) \\
&\leq B \sqrt{\frac{2H \ln 2 + 2 \ln(1/\delta)}{m}} + \theta\sqrt{2}\gamma_1 + \gamma_2. \qquad (3.18)
\end{aligned}
$$

Eq. (3.17) is due to the triangle inequality. The first term of Eq. (3.18) is because B is the upper bound of the loss function and because of the Bretagnolle-Huber-Carol inequality (3.1), and the latter is due to the robustness of the problem. □

It is worth noting that this bound tends to zero as the covering number H increases ($\gamma_1 \to 0$ and $\gamma_2 \to 0$) and the number of samples $m \to \infty$. In the following subsections, we instantiate $s_k(.)$ with two different metric functions: first as a Mahalanobis-like distance and then as a bilinear similarity.

3.3.2 Derivation for Mahalanobis-like Local Metrics

The Mahalanobis distance of a pair (x_1, x_2) valued for a local model k can be written as $s_k(x_1, x_2) = d_{A_k}(x_1, x_2) = \sqrt{(x_1 - x_2)^T A_k (x_1 - x_2)}$ with A_k the corresponding (learned) PSD matrix whose Cholesky decomposition is $A_k = L_k^T L_k$. Thus, our objective function takes the following form:

$$
\begin{aligned}
F_S(W) = \frac{1}{m} \sum_{i=1}^{K} \sum_{j=1}^{i} \sum_{z \in R_{ij}} &\left| \sum_{k=1}^{K} W_{ijk} d_{A_k}(x_1, x_2) - y(x_1, x_2) \right| \\
&+ \lambda_1 D(W) + \lambda_2 S(W) \qquad (3.19)
\end{aligned}
$$

where $A = \{A_1, .., A_K\}$ is a set of Mahalanobis metrics.

Lemma 3.3 *$\forall k = 1, ..., K$ the Mahalanobis distance $d_{A_k}(x_1, x_2)$ is θ_k-Lipschitz w.r.t. the norm $\|.\|_2$, with $\theta_k = \sqrt{2}\,\|L_k\|_2$.*

Proof. See Appendix C. □

Lemma 3.4 *$F_S(W)$ is $(H, 2\gamma_1\,\|L\|_2 + \gamma_2)$-robust and for any $\delta > 0$ with probability at least $1 - \delta$, we have:*

$$|R_{\mathcal{D}}(W) - \hat{R}_S(W)| \leq 2\gamma_1\,\|L\|_2 + \gamma_2 + B\sqrt{\frac{2H\ln 2 + 2\ln 1/\delta}{m}}. \tag{3.20}$$

The constant $\|L\|_2$ corresponds to $\max_{k=1,...,K}\|L_k\|_2$ so that $\theta = \sqrt{2}\,\|L\|_2$, because $\theta_k = \sqrt{2}\,\|L_k\|_2$.

3.3.3 Derivation for Local Bilinear Similarities

The bilinear similarity of a pair (x_1, x_2) can be written as $s_k(x_1, x_2) = x_1^T A_k x_2$. Thus, our problem becomes:

$$F_S(W) = \frac{1}{m}\sum_{i=1}^{K}\sum_{j=1}^{i}\sum_{z\in R_{ij}}\left|\sum_{k=1}^{K} W_{ijk}x_1^T A_k x_2 - y(x_1, x_2)\right| \\ + \lambda_1 D(W) + \lambda_2 S(W) \tag{3.21}$$

where $A = \{A_1, .., A_K\}$ is a set of bilinear similarities. Recalling that $\forall x \in \mathcal{X}$, $\|x\| \leq R$, with $R \in \mathbb{R}^+$:

Lemma 3.5 *$\forall k = 1, ..., K$ the bilinear similarity $s_k(x_1, x_2) = x_1^T A_k x_2$ is θ_k-Lipschitz w.r.t. the norm $\|.\|_2$, with $\theta_k = \sqrt{2}\,\|A_k\|_2\,R$.*

Proof. See Appendix C. □

Lemma 3.6 *$F_S(W)$ is $(H, 2\gamma_1\,\|M\|_2\,R)$-robust and for any $\delta > 0$ with probability at least $1 - \delta$, we have:*

$$|R_{\mathcal{D}}(W) - \hat{R}_S(W)| \leq 2\gamma_1\,\|M\|_2\,R + \gamma_2 + B\sqrt{\frac{2H\ln 2 + 2\ln 1/\delta}{m}}. \tag{3.22}$$

$\|A\|_2 = \max_{k=1,...,K}\|M_k\|_2$ so that $\theta = \sqrt{2}\,\|A\|_2\,R$, because $\theta_k = \sqrt{2}\,\|A_k\|_2\,R$.

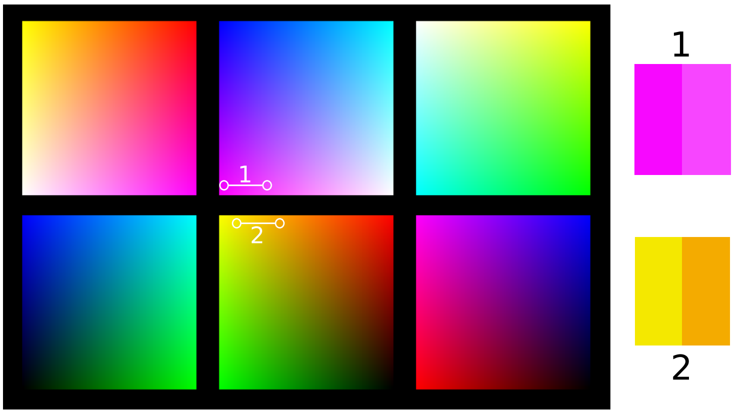


Figure 3.6 – *Representation of the six faces of the RGB cube. Two pairs of colors at the same Euclidean distance are selected, though the upper pair is perceptually closer than the lower one.*

3.4 Experiments

In this section, we aim at showing that **C2LM** is well suited to deal with both distance and similarity functions. Therefore, we empirically evaluate our method on two applications: first on the estimation of perceptual color distances and then on the estimation of semantic similarities between words.

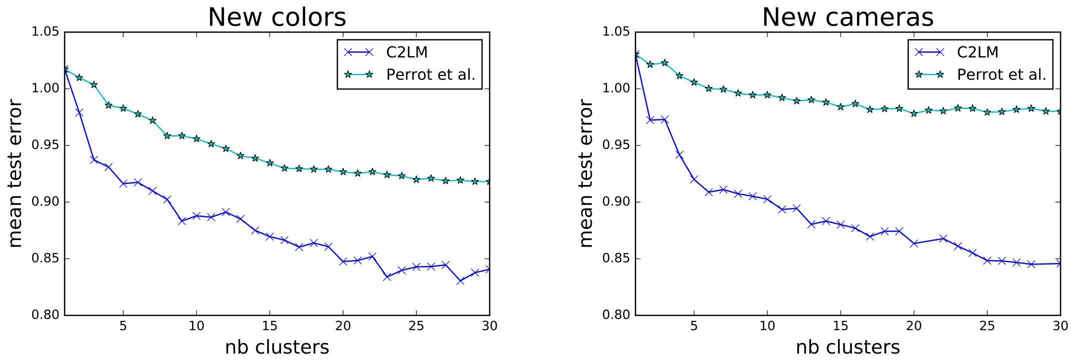
3.4.1 Applications and Datasets

Modeling perceptual color distances It is well known that a human observer cannot distinguish all the shades corresponding to the different mixtures of light wavelengths. We are more sensitive to medium wavelengths (to green/yellow colors) than to short and large wavelengths of the visible spectrum. Moreover, human perception strongly depends on variations of visual conditions, such as brightness, luminance, background changes, and so on. The perceived difference between colors cannot be modeled using an additive color space as the RGB space, because the corresponding distance is not proportional to the Euclidean distance on that space (see Figure 3.6).

Because numerous Computer Vision tasks can benefit from a good perceptual color distance, such as image segmentation, object detection and tracking, a lot of work has been put into finding a way of estimating it. In the past, several perceptual color spaces have been proposed to better model the human color perception: CIELuv and CIELab (see [184]) are two examples of such efforts to model uniform perceptual spaces. However, these spaces are still sensitive to some visual variations and can only be used under standard image acquisition conditions. This is because camera configurations, such as white balance, demosaicing and gamma correction, have a huge impact on the final perception of the color distances.

C2LM is particularly suited to the task of modeling a perceptual color distance that is invariant to acquisition conditions, by learning the relationships between variations in visual conditions and final perceived distance. For our experiments, we use the dataset built by Perrot et al. [156]. The dataset consists of 29580 color patches, expressed in their RGB coordinates and uniformly distributed on the RGB cube, and 41800 pairs of color patches, taken under several viewing conditions and with 4 different cameras, with their reference perceptual distance ΔE_{00}. Such a target distance corresponds to the perceptual color distance and has been computed using the CIEDE2000 color-difference formula [167] based on CIELab space, under controlled conditions. However, it is reliable only under standard viewing conditions (illuminant D65, illuminance of 1000 lx, etc. defined by the International Commission on Illumination CIE) so it cannot be used in all circumstances. We propose, here, to approximate the true perceptual distance between two colors that is independent of the viewing conditions. For this aim, the color patches are clustered using k-means (using the Euclidean distance on the RGB space) and on each so-found region a local model is learned as a Mahalanobis-like distance (using the color pairs whose patches both belong to that region). We then apply our method for learning linear combinations of those distance functions with manifold regularization, as detailed in Section 3.2. We compare our method to [156], where the authors learn a set of Mahalanobis-like metrics independently from each other: they cluster the color patches using k-means and learn a local metric on each cluster and a global one with the color pairs whose patches belong to different clusters; they compute the distance between two colors using the local distance if they belong to the same cluster or the global distance if they do not. As [156], we evaluate our method on two different tasks (testing on unseen colors and on color pairs from unseen cameras).

Modeling semantic similarities The semantic similarity between words is defined as the measure of closeness in meaning between two terms. It is a measure defined by human perception and it cannot be expressed by exact rules. Nevertheless, it can be estimated by representing the words as vectors of a continuous space (word embedding) and computing their distance or similarity, for instance the Euclidean distance or the cosine similarity. We show how a word embedding can be enhanced using our method. As in the previous application, we learn a local model on each cluster of words (the clustering procedure accomplished using k-means with the Euclidean distance on the word embedding) and then we apply **C2LM** on the



(a) *Results on unseen colors.*

(b) *Results on unseen cameras.*

Figure 3.7 – *Comparison of our method and local metric learning approaches on the application of perceptual color distance estimation. The used criterion is the mean loss over the test instances.*

learned local models, which, in this case, are bilinear forms (see 3.3.3) computed independently using the following optimization problem:

$$\arg\min_{B_k} \frac{1}{m} \sum_{z \in R_{kk}} \left| x_1^T B_k x_2 - y(x_1, x_2) \right| + \|B_k\|_{\mathcal{F}}. \tag{3.23}$$

For our experiments, we extracted the word embedding from the Reuters News stories[2] text corpus using the Hellinger PCA as presented in [111]. We then evaluate different methods on the WordSim353-similarity dataset: it is composed of 353 pairs of english words and for each pair we have at our disposal its semantic similarity as estimated by a human expert. We will compare our method with computing the cosine similarity directly on the embedding and with learning a set of local bilinear similarities and a global one. Because the cosine similarity is capable of predicting scores only in the interval $[-1, 1]$ and the similarity scores of the dataset are between 0 and 10, we first rescaled the target scores into the interval $[-1, 1]$.

3.4.2 Implementation and Results

We implemented our algorithm using the Cvxpy library[3] and its SCS solver (see [148]). The code is available at https://gitlab.univ-st-etienne.fr/vzantedeschi/c2ml/tree/master. For our experiments, we computed the best values for parameters λ_1 and λ_2 executing a grid search hyperparameter optimization by cross-validation: we fixed them to $\lambda_1 = 0.01$ and $\lambda_2 = 10000$ for the first application and to $\lambda_1 = 0.0001$ and $\lambda_2 = 100$ for the second one.

[2] http://about.reuters.com/researchandstandards/corpus/

[3] cvxpy.readthedocs.org/en/latest/

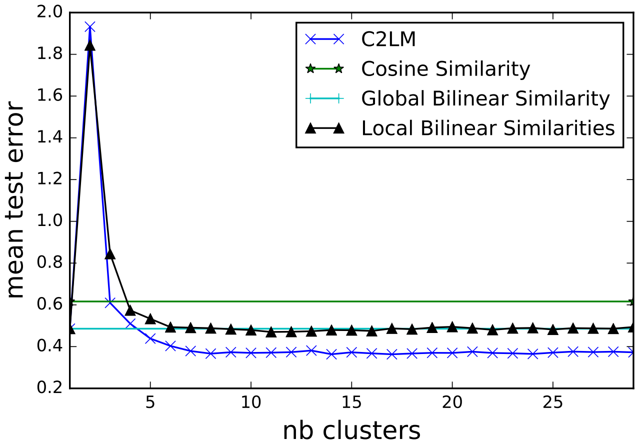


Figure 3.8 – *Comparison of our method and local metric learning approaches on the application of semantic similarity estimation. The used criterion is the mean loss over the test instances.*

In Figure 3.7, we represent the variation of the test loss over the number of clusters for the two tasks. For the application on unseen colors, we show the mean results of a 6-fold cross validation of the color patches set, obtained from five iterations. We notice that as the number of clusters increases the empirical test loss decreases: a set of local metrics captures the underlying geometry of the color space much better than a unique global metric ($K = 1$). Moreover, with a small number of clusters, the learned linear combinations are more expressive than the local metrics: thanks to the prior influence and similarity regularizations, we successfully prevent the model from over-fitting the training instances. This trend is more and more prominent as the number of clusters grows. For the application on unseen cameras, we report the mean results of a 4-fold cross validation (leave one camera out) of the color pairs set, iterated 3 times. Once again, our method outperforms the state of the art. For both tasks, we can note that with a very limited number of clusters, that is only 5, our test loss is always smaller than every test loss the approach of [156] could attain, even with 30 clusters.

Concerning the application on semantic similarity, Figure 3.8 presents the mean results of a 6-fold cross validation, iterated five times. We can note that learning metrics on the word embedding gives better results than applying directly the cosine similarity, but also that the local metrics fail to improve the test error with respect to a global bilinear form. On the contrary, **C2LM** converges with a limited number of clusters to an enhanced test error. We also notice that, against the trend, the test error increases when passing from one to two clusters. This can be explained by the fact that the quality of the local models is so poor that the learned convex combinations of them cannot be good.

3.4.3 Illustration of Learned Combinations

In this Section, we illustrate a metric learned using **C2LM** and compare it with the one learned using a local metric learning approach.

In the context of the perceptual color distance, Figure 3.9 shows a 2D projection of the contour lines of our learned combination of metrics, drawn around an arbitrary point, in the RGB space. While using only local models causes a strong discontinuity at the boundaries of the cluster (because one jumps from a local metric to the global one), we can see that our learned metric is smoother. In addition, it is evident that, while comparing points that do not belong to the same region, our metric is more accurate because our method captures better geometric variations of the space than a global linear metric.

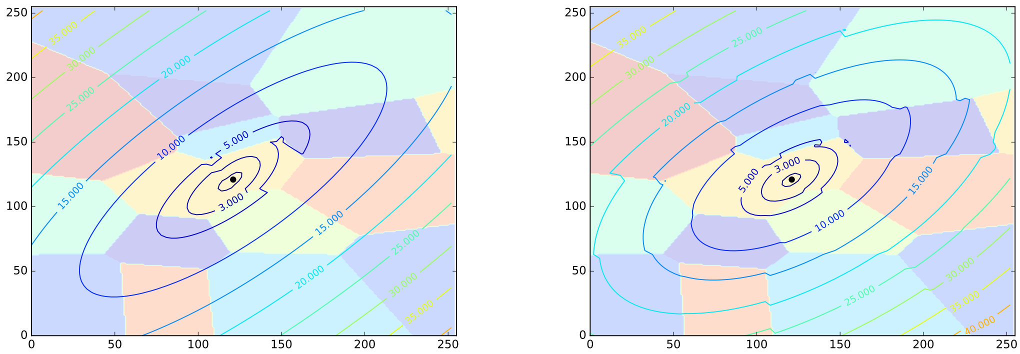


Figure 3.9 – *On the left: contour lines obtained using [156]'s method; on the right: contour lines obtained using* ***C2LM****.*

3.5 Conclusion

In this chapter, we proposed a new method for enhancing local metric learning based on a spatial partitioning of the inputs. We learned convex combinations of local models given prior knowledge on their correlations, in order to attain smoothness in the predictions and to avoid over-fitting. We proved that our learning algorithm is theoretically founded w.r.t. the algorithmic robustness framework. Empirically, we showed how our method improves the results on two different problems w.r.t. both global approaches and local ones, based on data partitioning.

A direct extension of the method would be to learn simultaneously both the local metrics and their linear combinations. The optimization problem would take the form of a double regression, one over the points belonging to the same region and one for all the others. In this way, we could guarantee that the local models perform well both locally and globally speaking by means of regularization and it would be possible to derive a tighter generalization bound.

In the next chapter, we propose another local learning approach exploiting a partition of the input data. However, we will tackle the problem of learning local models from another perspective. We will consider a partition of the inputs that is not based on spatial criteria, but on metadata, such as the user generating the data. In addition, we will focus on a setting more constrained in terms of resources, which will be limited, and privacy, because we will work with potentially sensitive data.

Boosting Personalized Models 4

This chapter is based on the publication

Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. Decentralized Frank-Wolfe boosting for collaborative learning of personalized models. In *CAp*, 2018.

In this chapter, we focus our work on learning local models on data partitioned using a criterion other than spatiality. More precisely, we consider data generated by a set of agents, that has been collected by their personal devices so that is naturally clustered in subsets. Like in the previous chapter, we aim to learn a local model per agent (using its local dataset), that we will denote hereafter as personalized. Furthermore, in order to improve the generalization ability of the optimized models, we add smoothing constraints based on a user similarity criterion. However, unlike for **C2LM**, we optimize all personalized models in a joint optimization problem taking into account both local optimality and overall smoothness. Moreover, learning from personal data raises new challenges that we need to take into account: the data collected by each user is extremely large and is potentially sensitive. For these reasons, we consider a new learning paradigm that is particularly suited to our scenario. Decentralized learning comprehends all techniques working on a non-hierarchical graph of users, in which each user keeps its data on-site and communicates with its neighbors only its model updates. Our method exploits this decentralized architecture to learn personalized models in a collaborative manner on a graph which reflects the similarities between users. As we will illustrate throughout the chapter, we formulate our problem as a graph-regularized l_1-Adaboost and optimize it using Frank-Wolfe algorithm, in order to get expressive models with minimal communication complexity. To make up for the potential absence of background knowledge on the similarities between users, we additionally introduce a formulation to jointly learn the personalized models and the graph topology through an alternating optimization procedure.

The Chapter is organized as follows: in Section 4.1, we give an overview of decentralized learning and its state of the art techniques; Section 4.3 introduces our problem formulation and the proposed decentralized algorithm; in Section 4.4, we analyze the convergence rate, memory consumption and communication complexity of our algorithm; Section 4.5 extends the previous framework by allowing us to optimize the collaboration graph together with the classifiers; finally, we empirically compare its performance to state-of-the-art decentralized techniques in Section 4.6.

4.1 New challenges in data collection

In the era of big data, the classical paradigm is to build huge data centers to collect and process users' data. This centralized access to resources and datasets simplifies some procedures, such as building predictive models with machine learning, but it comes with major drawbacks. From the company's point of view, the need to gather and analyze the data centrally induces high infrastructure costs. Furthermore, as the server represents a single point of entry, one needs to ensure that it is secure enough to prevent attacks that could put the entire user database in jeopardy. In this respect, the recent GDPR European regulation imposes responsibility and accountability: companies are expected to develop costly security measures and to design private-by-design systems. On the user end, the drawbacks include limited control over their own data as well as possible privacy risks, which may come from the aforementioned attacks but also from potentially loose data governance policies on the part of the companies. A more subtle problem is the risk of being trapped in a "single thought" model which fades individual users' specificities.

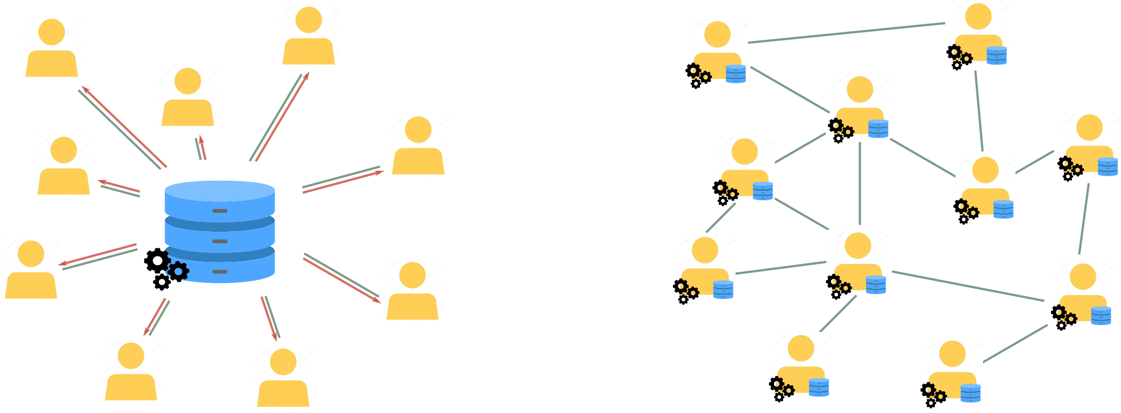


Figure 4.1 – *In Centralized learning (on the left) all agents are connected to a central node where their data is collected and processed. Conversely, in Decentralized learning (on the right), the users are organized in a non-hierarchical graph where only model-updates are passed between nodes.*

For all these reasons and thanks to the advent of personal devices and their ever increasing computational power and storage capacity, we are currently witnessing a shift from the above centralized paradigm to a more decentralized one (see

Figure 4.1). For instance, modern programming frameworks delegate large parts of their code execution to the local devices, reducing communication costs and allowing better personalization.[1] In the context of machine learning, this shift translates into keeping the data on the users' devices and leveraging their resources to train the predictive models in a collaborative manner. This requires a compromise on a key assumption of more traditional distributed learning approaches, namely that the data is balanced and uniformly distributed across machines [4, 13, 127, 166, 225, 226], which is not realistic for local datasets generated by a diverse set of users.

4.1.1 Decentralized Learning Approaches

Traditional distributed approaches [4, 8, 13, 127, 166, 172, 225, 226] utilize a network of workers for learning on large-scale datasets. By exploiting the computational resources of the cluster of machines, such techniques offer ways of speeding up computations, and possibly avoiding storage issues. This is achieved by coordinating the machines to work in parallel on mini-batches of data. Generally, these approaches make the assumption that data is evenly distributed over the network and is i.i.d.. As a consequence, they are by-design incapable of dealing with datasets like the ones generated by networks of users. As a matter of fact, in these datasets, each user's sample is biased by the personal generation procedure, thus is not representative of the global data distribution.

There are two main lines of work to cope with this kind of data. In federated learning [98, 99, 131], one relies on a coordinator-clients architecture. Each device (*client*) computes an update of the current global model based on its local data and sends it to the server (*coordinator*), which aggregates the updates in a way that tries to correct for discrepancies in the local datasets, and broadcasts the result back to the clients. In practice, the dependence on the coordinator creates a single point of failure as well as a communication bottleneck when the number of clients is large [119]. Decentralized learning aims to fix this problem by removing the coordinator and relying only on local peer-to-peer exchanges [45, 57, 119, 142, 180, 202, 224]. In both federated and decentralized learning, achieving low communication is key to the practical efficiency of the algorithms (especially when running on small devices), and a lot of effort has been dedicated towards reducing the communication complexity while preserving accuracy [99, 177, 181, 226].

[1] See for instance React.js.

4.1.2 Learning Personalized Models

Another promising and complementary direction to further account for the differences in the users' datasets is to learn personalized models instead of assuming that there exists a single global model which is accurate for all users. The hope is to be able to improve the user experience by modeling its specific taste and data distribution. The decentralized architecture is particularly suited for learning personalized models: it allows us to extract reliable learners without gathering sensitive data on a single server, reducing the risks of data leakage; it can be easily powered with privacy-preserving protocols; it can be deployed for learning a personal model per network node, instead of a unique global one. Moreover, if the network of nodes is constructed making its topology reflect the similarities between users, the personalized models can be optimized in a collaborative way, taking advantage of information coming from similar users. This has been explored both in federated [171] and decentralized [14, 191] settings by regularizing personalized models to be close for "similar" users so as to improve generalization (in particular for users with small datasets). Note that choosing appropriate similarity scores between users is often difficult, especially in the decentralized setting. Another limitation of the above personalized approaches is that they consider only linear models, and it is not clear how to extend them to nonlinear models while retaining their convergence properties.

4.2 Related work

In this section, we discuss in more details the two lines of previous works that are most closely related to our approach, namely distributed boosting and distributed training of personalized models.

Distributed Boosting. As highlighted in Chapter 2, boosting methods are principled and powerful ways to construct a nonlinear classifier by adaptively combining base functions. Several distributed versions of Adaboost have been proposed to tackle the setting where data is partitioned across a set of machines [47, 110, 200]. The work of [110] proposes to select base hypotheses on each machine independently and to combine them at each round using a voting scheme. Improved algorithms that are less prone to overfitting the dataset of a single machine have been recently introduced by [47]. Like in our approach, the method of [200] operates on a general graph and exchanges information across neighbors, but assumes that agents can subsample from a global dataset and the main goal is to provide more robust solutions by combining solutions trained on bootstrap samples. There has also been some work on distributed Frank-Wolfe algorithms which can be used to solve the l_1-Adaboost problem. The approach of [18] is communication-efficient but can only be applied to the case where base functions (not data points) are distributed across machines, while [108] is based on averaging gradients and is hence communication-intensive for boosting with many base functions. All the above approaches learn a single global classifier, while we focus on learning personalized models.

In [172], the authors propose an approach to learn personalized classifiers with boosting for activity classification in microblogs. The network graph is given by a social network, and the model of each user is regularized to make similar predictions to that of its neighbors. Their approach has several drawbacks compared to ours. First, agents need to share their personal dataset with their neighbors, which can be costly in terms of communication and may be undesirable due to privacy concerns. Second, the convergence of their procedure is only to a local optimum, without an established rate. Finally, they assume the availability of a relevant social network graph.

Distributed Learning of Personalized Models. Distributed Multi-Task Learning (MTL) has been considered in several recent works [12, 117, 196, 197], but these typically consider a small number of tasks with well-balanced data across tasks, and often focus on learning linear models. Recent work on federated and decentralized learning algorithms [14, 171, 191] has demonstrated how fostering model smoothness for similar tasks increases the generalization performance in the presence of dataset imbalance across tasks. The federated learning approach [171] learns personalized models as well as the collaboration affinities by alternating between a local step where agents optimize their own model and an aggregation step where the similarities between tasks are updated based on the current personalized models. However, it is limited to linear models and relies on a coordinator node. In contrast, [14, 191] operate in the decentralized setting and propose algorithms based on the Alternating Direction Method of Multipliers (ADMM) [31] and block coordinate descent which can scale to many agents. The high-level idea is to alternate between local model updates and weighted model averaging with neighbors. They however assume that the collaboration graph is known a priori. It is worth noting that all these approaches have a communication cost per iteration which scales linearly with the number of model parameters (which corresponds to the data dimension for linear models). In contrast to the previous work, our approach is able to learn nonlinear models which can capture complex structure in the data with low communication cost, while also learning the collaboration graph. We believe that this combination of features greatly improves the applicability of the framework.

4.3 Decentralized Frank-Wolfe boosting for learning personalized models

In this chapter, we propose a decentralized method for optimizing personalized nonlinear models for classification with low communication cost. We first introduce a novel problem formulation based on l_1-Adaboost [68, 168], which allows to build nonlinear classifiers as combinations of a set of base functions. We achieve collaboration and personalization by introducing a trade-off between (i) making the model of an agent accurate on its local dataset, and (ii) selecting base functions that are popular among the agent's neighbors.

We then propose a decentralized optimization algorithm based on the Frank-Wolfe algorithm [65], building upon the work of [195] for boosting a global model in the centralized setting. At each iteration, a random agent wakes up and greedily updates its personalized model by incorporating a single base function at a time. The sparsity of these updates enables very low communication costs (logarithmic in the number of base functions). Finally, as designing an appropriate collaboration network topology is difficult and often requires side information about the application of interest, we propose a strategy to learn this topology along with the personalized classifiers through an alternating optimization scheme.

4.3.1 Notations

We consider a set of K agents, each with a different binary classification task. Each agent k holds its own labeled dataset S_k of size m_k. For convenience, we denote the total number of samples by $m = \sum_{k=1}^{K} m_k$ and the union of the local datasets by $S = \cup_{k=1}^{K} S_k = \{(x_i, y_i)\}_{i=1}^{m}$, where observations $x_i \in \mathcal{X} \subset \mathbb{R}^d$ are associated with a binary label $y_i \in \{-1, 1\}$.

All agents have local access to the same set of n real-valued base functions $H = \{h_j : \mathcal{X} \to \mathbb{R}\}_{j=1}^{n}$. Each agent k aims at learning a binary classifier in the form of a linear combination of the base functions in H, i.e. a mapping $x \mapsto \text{sign}[\sum_{j=1}^{n} (\alpha_k)_j h_j(x)]$ parameterized by a weight vector $\alpha_k \in \mathbb{R}^n$. We will denote by $A \in \mathbb{R}^{m \times n}$ the matrix whose entry $a_{ij} = y_i h_j(x_i)$ gives the margin achieved by the base classifier $h_j \in H$ for the training sample $(x_i, y_i) \in S$. We will also use $A_k \in \mathbb{R}^{m_k \times n}$ to denote the margin matrix restricted to the samples in the dataset S_k of agent k, so that for $i \in [m_k]$, $(A_k \alpha_k)_i = y_i \sum_{j=1}^{n} (\alpha_k)_j h_j(x_i)$ gives the margin achieved by the classifier α_k on the i-th data point in S_k.

Instead of learning their classifiers on their own, agents will collaborate to learn better classifiers by exchanging information over a network represented by an undirected and weighted graph $\mathcal{G} = (V, E)$. Each edge weight W_{kl} reflects the similarity of the tasks associated with users k and l ($W_{kl} = W_{lk}$), with $W_{kl} = 0$ for $(v_k, v_l) \notin E$. We indicate by $D_k = \sum_{l=1}^{K} W_{kl}$ the degree of node k. In the rest of this section, we will consider these similarity weights to be given (we show in Section 4.5 how to learn them together with the classifiers). To ensure the scalability of our algorithms to a large number of agents, we will favor local communication schemes, i.e. each agent k will only need to send messages to its direct neighbors $N_k = \{l : (v_k, v_l) \in E\}$ in the network, which will typically be a small set compared to the number of nodes K.

As agents have datasets of different sizes, we introduce a confidence value $c_k \in \mathbb{R}^+$ for each agent k which should be thought of as proportional to m_k. This confidence score can be set for instance as $c_k = \frac{m_k}{\max_l m_l}$.

4.3.2 Graph Regularization Formulation

We now introduce our formulation as a joint optimization problem over the classifiers $\alpha_1, \ldots, \alpha_K$. It is essentially a personalized version of l_1-Adaboost [168] with additional graph regularization, and can be written as follows:

$$\min_{\|\alpha_1\|_1 \leq \beta, \ldots, \|\alpha_K\|_1 \leq \beta} f(\alpha_1, \ldots, \alpha_K) = \sum_{k=1}^{K} D_k c_k \log\left(\sum_{i=1}^{m_k} \exp\left(-(A_k \alpha_k)_i\right)\right) + \frac{\mu}{2} \sum_{k=1}^{K} \sum_{l=1}^{k-1} W_{kl} \|\alpha_k - \alpha_l\|^2. \tag{4.1}$$

The objective function in (4.1) is composed of two terms. The first one is a sum of Adaboost logistic loss functions (one per agent), involving only their personal model and their local dataset. Only optimizing this first term would amount to having each agent k learn its classifier α_k in isolation by minimizing the loss on its local dataset. The second term is a graph regularization term that enables collaboration by encouraging each agent k to select the same weak classifiers as its neighbor $l \in N_k$ when the edge weights W_{kl} are large. Notice that the confidence weight c_k in the first term adjusts the trade-off between these two terms differently for each agent depending on how much local data the agent holds (D_k has a strictly normalization role). The parameter μ is used to globally balance the two terms and β bounds the l_1 norm of the learned models.

Note that Problem (4.1) is convex. In particular, the objective function is convex and continuously differentiable, and the feasible domain is a compact and convex subset of $(\mathbb{R}^n)^K$ that we will denote by $\mathcal{M}$.

4.3.3 Decentralized Frank-Wolfe Algorithm

In this section, we propose an algorithm to solve (4.1) in the decentralized setting. This means that, in order to perform an update, an agent should only need to send messages to its direct neighbors in the network graph $\mathcal{G}$, which ensures that the procedure scales well to large networks. As standard in the literature (see for instance [33]), we consider that each agent k is equipped with a local clock that ticks independently and follows a 1-Poisson distribution. This is equivalent to considering a global clock (with counter t) which ticks each time one of the local clock ticks. When the local clock of agent k ticks, it will update its local classifier α_k then broadcasts the update to its neighborhood N_k. We assume that agents have access to the global counter t, hence some updates can occur in parallel in different parts of the network.

Our algorithm is based on Frank-Wolfe (FW) [65, 89], which has recently been applied to the l_1-Adaboost problem in the centralized and non-personalized settings [195]. The main idea of FW is to iteratively update the current solution by moving towards the minimizer of the linearized objective over the feasible domain. Specifically, when applied to our problem formulation (4.1) and restricting our attention to the model α_k of agent k, a Frank-Wolfe update takes the form $\alpha_k \leftarrow (1-\gamma)\alpha_k + \gamma s_k$ where $\gamma \in [0,1]$ is the step size,

$$s_k = \underset{\|s\|_1 \leq \beta}{\arg\min} \; \langle s, g_k \rangle, \tag{4.2}$$

and $g_k = \nabla_k f(\alpha_1, \ldots, \alpha_K) \in \mathbb{R}^n$ is the gradient at the current solution restricted to the k-th block of coordinates (corresponding to the model of agent k). Note incidentally that the FW update takes the form of a convex combination of two feasible points, hence the updated model remains feasible $\left(\left\|\alpha_k\right\|_1 \leq \beta\right)$.

It is well known that a solution of (4.2) is given by $\beta \operatorname{sign}(-(g_k)_{j_k}) e^{j_k}$ where $j_k = \arg\max_j (|g_k|)_j$ and e^{j_k} is the unit vector with 1 in the j_k-th entry. In other words, FW updates a single coordinate of the current model α_k which corresponds to the maximum absolute value entry of the gradient g_k. For Problem (4.1), the gradient is given by

$$g_k = -D_k c_k w_k^T A_k + \mu \left(D_k \alpha_k - \sum_l W_{kl} \alpha_l \right), \text{ with } w_k = \frac{\exp(-A_k \alpha_k)}{\sum_{i=1}^{m_k} \exp(-A_k \alpha_k)_i}. \tag{4.3}$$

The first term in g_k plays the same role as in standard Adaboost: the j-th entry (corresponding to the base classifier h_j) is larger when h_j achieves a large margin on the training sample S_k reweighted by w_k (points that are currently poorly classified get more weight). On the other hand, the more h_j is used by the neighbors of k, the larger the j-th entry of the second term. By applying the FW update (4.2),

Algorithm 1: Decentralized Frank-Wolfe Graph Regularized Boosting

```
Input: $A, S$
Initialize $\alpha_k^{(0)}$ s.t. $\left\|\alpha_k^{(0)}\right\|_1 \leq \beta$ for all $k \in [K]$
for $t = 1$ to $T$ do
    $\gamma^{(t)} = \frac{2K}{t+2K}$
    pick $k$ uniformly from $[1 \dots K]$
    $w_k = \frac{\exp(-A_k \alpha_k^{(t-1)})}{\sum_{i=1}^{m_k} \exp(-A_k \alpha_k^{(t-1)})_i}$
    $g_k^{(t)} = \nabla_k f(\alpha_1^{(t-1)}, \dots, \alpha_K^{(t-1)})$
    $j_k^{(t)} = \arg\max_j (|g_k^{(t)}|)_j$
    $s_k^{(t)} = \beta \operatorname{sign}(-(g_k^{(t)})_j) e^{j_k^{(t)}}$
    $\alpha_k^{(t)} = (1 - \gamma^{(t)}) \alpha_k^{(t-1)} + \gamma^{(t)} s_k^{(t)}$
end for
```

we are able to preserve the flavor of boosting (incorporating a single base function at a time which performs well on the reweighted sample) with an additional bias towards the selection of base classifiers which are popular among neighbors. Note that the relative importance of the two terms depends on the agent confidence c_k.

Building upon the above ideas, we propose a decentralized algorithm to solve Problem (4.1), see Algorithm 1. All classifiers are initialized to some arbitrary feasible point (such as the zero vector). At each iteration (corresponding to a tick of one of the local clocks), a single agent becomes active and performs a FW update on its local model based on its local dataset and the current models of its neighbors. This corresponds to optimizing at each iteration a randomly picked block α_k (drawn uniformly) of the whole set of parameters, as done in [106]. After the update, the agent broadcasts its new model to its neighbors. Thanks to the sparsity of the FW updates, we will see in Section 4.4.2 that this can be done with very low communication cost.

4.4 Analysis

4.4.1 Convergence Analysis

The convergence analysis of our algorithm follows the proof technique proposed by Jaggi in [89] and refined by [106] for the case of block coordinate Frank-Wolfe. We start by introducing some useful notations related to our problem (4.1) and by defining key quantities for the analysis. For any $k = 1, \dots, K$, we let $\mathcal{M}^k = \{\alpha_k \in \mathbb{R}^n : \left\|\alpha_k\right\|_1 \leq \beta\}$ and denote by $\mathcal{M} = \mathcal{M}^1 \times \cdots \times \mathcal{M}^K$ our feasible domain in (4.1). For convenience, we will use $\alpha = [\alpha_1, \dots, \alpha_K] \in (\mathbb{R}^n)^K$ to denote the concatenation of the parameters of local classifiers $\{\alpha_k\}_{k=1}^K$ and refer to the objective function (4.1) as $f(\alpha)$. We also denote by $v_{[k]} \in \mathcal{M}$ the zero-padding of any vector $v_k \in \mathcal{M}^k$. Finally, for conciseness of notations, for a given $\gamma \in [0,1]$ we use $\hat{\alpha}$ to denote $\alpha + \gamma(s_{[k]} - \alpha_{[k]})$ and $\hat{\alpha}_k$ to denote $(1-\gamma)\alpha_k + \gamma s_k$.

In [89], the authors show that the dual gap defined as

$$\begin{aligned} \text{gap}(\alpha) &= \max_{s \in \mathcal{M}} \langle \alpha - s, \nabla(f(\alpha)) \rangle \\ &= \sum_{k=1}^K \text{gap}_k(\alpha_k) = \sum_{k=1}^K \max_{s_k \in \mathcal{M}^k} \big\langle \alpha_k - s_k, \nabla_k f(\alpha) \big\rangle \end{aligned} \tag{4.4}$$

can serve as a certificate of the quality of a current approximation of the optimum of the objective function. In particular, one can show that $f(\alpha) - f(\alpha^*) \leq \text{gap}(\alpha)$ where α^* is a solution of (4.1). The convergence of Frank-Wolfe is then established by showing that the surrogate gap cannot stay large over many iterations, because at a given iteration t of Algorithm 1 the block-wise surrogate gap at the current solution $\text{gap}_k(\alpha_k^{(t)})$ is minimized by the greedy update $s_k^{(t)} \in \mathcal{M}^k$.

To prove the convergence, the objective function needs to satisfy a form of smoothness expressed by a notion of curvature. More precisely, the global product curvature constant $C_f^{\otimes}$ of f over $\mathcal{M}$ is the sum over each block of the maximum relative deviation of f from its linear approximations over the block [106]:

$$C_f^{\otimes} = \sum_{k=1}^K C_f^k = \sum_{k=1}^K \sup_{\substack{\alpha \in \mathcal{M}, s_k \in \mathcal{M}^k \\ \gamma \in [0,1]}} \frac{2}{\gamma^2} \left(f(\hat{\alpha}) - f(\alpha) - \langle \hat{\alpha}_k - \alpha_k, \nabla_k f(\alpha) \rangle \right). \tag{4.5}$$

Each partial curvature constant C_f^k is upper bounded by the (block) Lipschitz constant of the partial gradient $\nabla_k f(\alpha)$ times the squared diameter of the block $\mathcal{M}^k$ [106]. The next lemma gives a bound on the product space curvature $C_f^{\otimes}$.

Lemma 4.1 *For Problem (4.1), we have* $C_f^{\otimes} \leq 4\beta^2 \sum_{k=1}^K \left(D_k c_k \left\| A_k \right\|_1^2 + \mu D_k \right)$.

Proof. See Appendix D. □

We now prove the convergence of Algorithm 1.

Theorem 4.1 *Algorithm 1 takes at most* $\frac{6K(C_f^{\otimes} + p_0)}{\varepsilon}$ *iterations to find an approximation of the optimum of problem (4.1) that satisfies* $f(\alpha) - f^* \leq \varepsilon$*, where* $p_0 = f(\alpha^{(0)}) - f^*$ *is the initial sub-optimality gap.*

Proof. Using the definition of the curvature (4.5) and rewriting $\hat{\alpha}_k - \alpha_k$ as $-\gamma_k(\alpha_k - s_k)$, we obtain

$$f(\hat{\alpha}) \leq f(\alpha) - \gamma \big\langle \alpha_k - s_k, \nabla_k f(\alpha) \big\rangle + \gamma^2 \frac{C_f^k}{2}.$$

In particular, at any iteration t, the previous inequality holds for $\gamma = \gamma^{(t)} = \frac{2K}{t+2K}$, $\alpha^{(t+1)} = \alpha^{(t)} + \gamma^{(t)}(s_k^{(t+1)} - \alpha_k^{(t+1)})$ with $s_k^{(t+1)} = \arg\min_{s\in\mathcal{M}^k}\langle s, \nabla_k f(\alpha^{(t)})\rangle$ as defined in Algorithm 1. Therefore, $\langle \alpha_k - s_k, \nabla_k f(\alpha)\rangle$ is by definition the gap $\text{gap}_k(\alpha_k)$ and

$$f\left(\alpha^{(t+1)}\right) \leq f\left(\alpha^{(t)}\right) - \gamma^{(t)}\text{gap}_k\left(\alpha_k^{(t)}\right) + \left(\gamma^{(t)}\right)^2 \frac{C_f^k}{2} \ .$$

By taking the expectation over the random choice of $k \sim \mathcal{U}(1, K)$ on both sides, we obtain

$$\begin{aligned} \mathbb{E}_k\left(f\left(\alpha^{(t+1)}\right)\right) &\leq \mathbb{E}_k\left(f\left(\alpha^{(t)}\right)\right) - \gamma^{(t)}\mathbb{E}_k\left(\text{gap}_k(\alpha_k^{(t)})\right) + \frac{\left(\gamma^{(t)}\right)^2 \mathbb{E}_k\left(C_f^k\right)}{2} \\ &\leq \mathbb{E}_k\left(f\left(\alpha^{(t)}\right)\right) - \frac{\gamma^{(t)}\text{gap}(\alpha^{(t)})}{K} + \frac{\left(\gamma^{(t)}\right)^2 C_f^{\otimes}}{2K} \ . \end{aligned} \tag{4.6}$$

Let us define the sub-optimality gap $p(\alpha) = f(\alpha) - f^*$ with f^* the optimal value of f. By subtracting f^* from both sides in (4.6), we obtain

$$\mathbb{E}_k\left(p\left(\alpha^{(t+1)}\right)\right) \leq \mathbb{E}_k\left(p\left(\alpha^{(t)}\right)\right) - \frac{\gamma^{(t)}}{K}\mathbb{E}_k\left(p\left(\alpha^{(t)}\right)\right) + \left(\gamma^{(t)}\right)^2 \frac{C_f^{\otimes}}{2K} \tag{4.7}$$

$$\leq \left(1 - \frac{\gamma^{(t)}}{K}\right)\mathbb{E}_k\left(p\left(\alpha^{(t)}\right)\right) + \left(\gamma^{(t)}\right)^2 \frac{C_f^{\otimes}}{2K}. \tag{4.8}$$

Inequality (4.7) comes from the definition of the surrogate gap (4.4) which ensures that $\mathbb{E}_k\left(p(\alpha)\right) \leq \text{gap}(\alpha)$.

Therefore, we can show by induction that the expected sub-optimality gap satisfies $\mathbb{E}_k\left(p(\alpha^{(t+1)})\right) \leq \frac{2K(C_f^{\otimes}+p_0)}{t+2K}$, with $p_0 = p\left(\alpha^{(0)}\right)$ as the initial gap. This shows that the expected sub-optimality gap $\mathbb{E}_k\left(p(\alpha)\right)$ decreases with the number of iterations with a rate $O(\frac{1}{t})$, which implies the convergence of our algorithm to the optimal solution. The final convergence rate can then be obtained by the same proof as [106] (Appendix C.3 therein). □

4.4.2 Communication and Memory Costs

We study the communication and memory costs of our algorithm for a generic graph $\mathcal{G} = (V, E)$ with K nodes and M edges. The following analysis stands for systems without failure (all sent messages are assumed to be correctly received). We express all the costs in number of bits, using Z to denote the bit length that is needed to represent a floating point number.

Memory Each agent needs to store its current model, a copy of its neighbors' models, and the similarity weights associated with its neighbors. Denoting by $(\#N_k)$ the number of neighbors of agent k, its memory cost is given by

$$Z\left(n + (\#N_k)(n+1)\right),$$

which leads to a total cost for the network of

$$KZ\left(n + \sum_{k=1}^{K}(\#N_k)(n+1)\right) = Z\left(Kn + 2M(n+1)\right).$$

The total memory is thus linear w.r.t. the number of edges M, the number of agents K and the number n of base classifiers.

Thanks to the sparsity of the updates of Algorithm 1, the dependency on n can be reduced from linear to logarithmic by representing models as sparse vectors. Specifically, when initializing the models to zero vectors, the model of an agent k who has performed t_k updates so far contains at most t_k nonzero elements and can be represented using $t_k(Z + \log n)$ bits: $t_k Z$ for the nonzero values and $t_k \log n$ for their indices.

Communication At each iteration, an agent k updates a single coordinate of its model α_k. Hence, it is enough to send to the neighbors only the index of the modified coordinate and its new value (or equivalently the index and the step size $\gamma_k^{(t)}$). Therefore, the communication cost of a single iteration is equal to $(Z + \log n)(\#N_k)$. This is in contrast to previous federated/decentralized approaches for learning personalized models: for these methods, the communication cost per iteration scales linearly with the number of parameters of the model.

After T iterations, the expected total communication cost is equal to

$$T(Z + \log n)\mathbb{E}_{k \sim \mathcal{U}(1,K)}\left(\#N_k\right) = \frac{2TM}{K}\Big(Z + \log n\Big).$$

Combining this with Theorem 4.1, the communication cost needed to obtain an approximation error smaller than ε is

$$\frac{12M(C_f + h_0)}{\varepsilon}\Big(Z + \log n\Big).$$

4.5 Learning the collaboration graph

Until now, we have assumed that the graph topology $\mathcal{G} = (V, E)$ and the weight matrix W describing the affinities between users were given. In practical applications,

W might be computed based on side information about the users (such as user profiles), but this is not always available or reliable. We propose an alternative strategy based on jointly optimizing a modified version of our problem (4.1) over α and W. In order to keep the algorithm decentralized and the communication costs low, we seek to learn a sparse graph. We note that the strategy proposed below can also be applied to the decentralized algorithms of [14, 191], which learn linear personalized models.

Notice that the graph regularization term (4.1) can be expressed as the trace of a quadratic form of the graph Laplacian matrix :

$$\sum_{k=1}^{K}\sum_{l=1}^{k-1} W_{kl}\|\alpha_k-\alpha_l\|^2 = tr(\alpha L \alpha^T)$$

where $L = D - W$ is the graph Laplacian and D the diagonal matrix of node degrees. Given a fixed D and some integer $q \geq 1$, we formulate the joint optimization problem as follows:

$$\begin{aligned} \min_{\alpha,W} \sum_{k=1}^{K} D_{kk} c_k \log\left(\frac{1}{m_k}\sum_{i=1}^{m_k}\exp\left(-(A_k\alpha_k)_i\right)\right) + \frac{\mu}{2} tr(\alpha(D-W)\alpha^T) \\ \text{s.t.} \begin{cases} W \geq 0,\ W^\top = W, \quad \forall l,k \in [K] : W_{kl} \leq \frac{D_{kk}}{q} \\ \forall\, k \in [K] : \quad \left\|\alpha_k\right\|_1 \leq \beta,\ \sum_{l=1}^{K} W_{kl} = D_{kk} \ . \end{cases} \end{aligned} \tag{4.9}$$

Observe that under the constraints in (4.9), the matrix $L = D - W$ is a proper graph Laplacian matrix and is thus positive semi-definite. In practice, we fix $D = I$ and solve (4.9) by optimizing alternatingy with respect to α and W, each of which is a convex subproblem. In the optimization over W, the constraint that all $W_{kl} \geq 1/q$ ensures that each agent is assigned at least q neighbors. As the objective function $tr(\alpha(D-W)\alpha^T)$ is linear in W, agents will tend to have exactly q neighbors with weights $1/q$, hence the learned graph can be understood as a q-regular graph.

All agents need to convene to jointly find the new weights. As this is a rather costly step, we perform this weight update only every t_w iterations of minimization over α. Hence we modify Algorithm 1 consequently. Every t_w iterations, all agents broadcast their model to all other agents and compute the new weights W locally. Then, given the new weights, the algorithm continues with the same Frank-Wolfe step size so as to avoid retraining the models from scratch.

This alternating optimization procedure is only guaranteed to converge to a local optimum of (4.9), hence initialization is important. For this reason, we initialize the weight matrix $W^{(0)}$ to the solution of (4.9) with α fixed to the purely local models (optimized independently on each agent). We, then, optimize (4.9) with respect to α given $W^{(0)}$ (starting from zero vector models) so that agents have sufficiently relevant neighbors right from the start.

4.6 Experiments

In this section we study the practical behavior of our method[2] and compare it to some baselines and state-of-the-art decentralized algorithms in terms of classification accuracy.

Synthetic dataset. To show the relevance of learning nonlinear models, we experiment with a synthetic dataset constructed from the classic two interleaving moons dataset. Inspired by [191], we construct a graph of K agents and randomly draw for each agent a rotation axis (coplanar to the Moons' points) from a Normal distribution. We generate the local datasets by drawing a random set of points (uniformly between 3 and 20) from the two Moons distribution for training and 100 for testing, while applying the random rotation to all points. We further add random label noise by flipping the labels of 5% of the training data.

We build a ground-truth collaboration graph where the similarities between agents are computed from the angle θ_{ij} between the agents' rotation axes, using $W_{ij} = \exp\left(\frac{\cos(\theta_{ij})-1}{\sigma}\right)$ with $\sigma = 0.1$. We drop all the edges with negligible weights. In order to increase the difficulty of the classification problems, all the points are embedded in $\mathbb{R}^D$ by adding random values for the $D-2$ empty axes. D is fixed to 20 in the experiments.

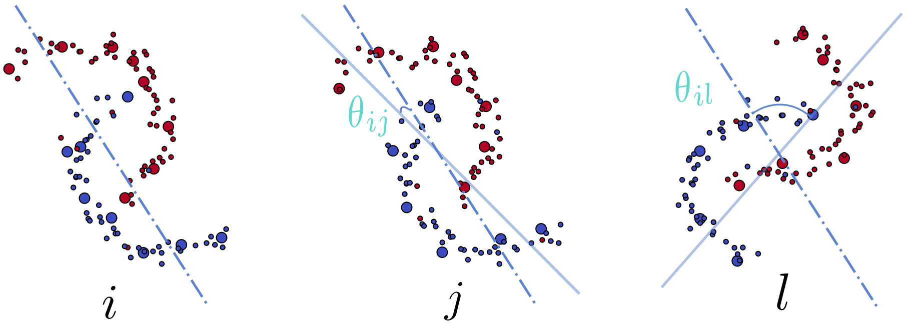
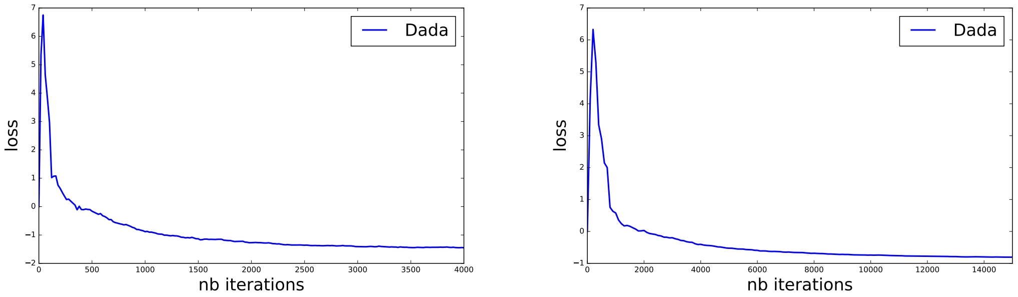


Figure 4.2 – *Illustration of the datasets of three users i, j, l. Each of them is rotated according to a rotation axis and the angle between users' rotation axes is retained to fix the ground-truth similarity. In this example, a larger similarity score is affected to the pair of users i, j than to i, l.*

Competitors We compare our algorithm (called **Dada** for Decentralized Adaboost) with several competitors which learn either global or personalized models, in either a centralized or a decentralized manner. We consider the following l_1-Adaboost variants. **Global l1-Adaboost** consists in learning a single global model

[2] Source code available at https://github.com/vzantedeschi/Dada



(a) $K = 20$. (b) $K = 100$.

Figure 4.3 – *The objective value for* ***Dada*** *w.r.t. number of iterations.*

over the joint dataset S through FW optimization. **Purely local models** independently learn K local models, through FW optimization over each local dataset. This algorithm learns personalized models, but without any collaboration between agents. **Global-local mixture** (inspired by the multi-task regularization of [41]) jointly learns K local models plus a global model optimized on the unified dataset S. The model associated with each node is then the sum of its purely local model and the global model. These methods use the same set of base functions, namely 200 decision stumps uniformly split between all D dimensions and value intervals.

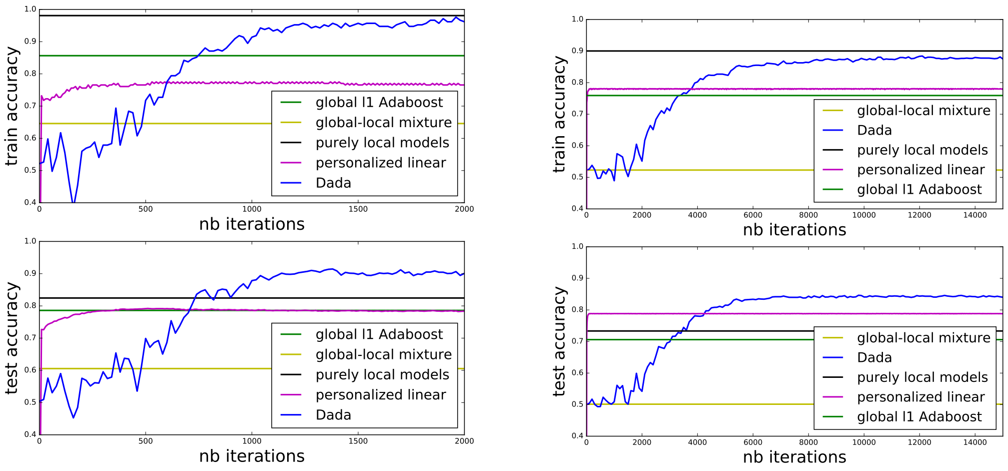
We also compare **Dada** to a decentralized method for collaboratively learning personalized linear models (**personalized linear**) described in [191] which is based on ADMM optimization [202]. This approach also relies on graph regularization.

In all the figures, we report the global accuracy and loss over all agents. Algorithms that build purely local or purely global models are shown with a straight horizontal line that corresponds to their value at convergence. We tune the hyper-parameters with 3-fold cross validation on the training agent datasets[3]. We apply a grid-search over the following set of values: β (l_1 norm constraint), common to all methods except for **personalized linear**, is searched for in $\{1, \dots, 10^4\}$. The parameters μ of **Dada** and **personalized linear** are searched for in $\{10^{-3}, \dots, 10^3\}$. The models are initialized to zero vectors unless specified otherwise.

4.6.1 Comparison with Competitors

In this first experiment, we study the convergence and the prediction accuracy of our method using the ground-truth collaboration graph. Figure 4.3 confirms that our method converges. As expected and as predicted by the theory (Theorem 4.1), the number of iterations needed to converge increases with the number of agents K. This is compensated by the fact that more decentralized updates can be done in parallel when the network is larger. Figure 4.4 shows the evolution of the training

[3] We observe that $\beta = 10$ and $\mu = 1$ for all models.



(a) $K = 20$.

(b) $K = 100$.

Figure 4.4 – *Training and test accuracy w.r.t. number of iterations.*

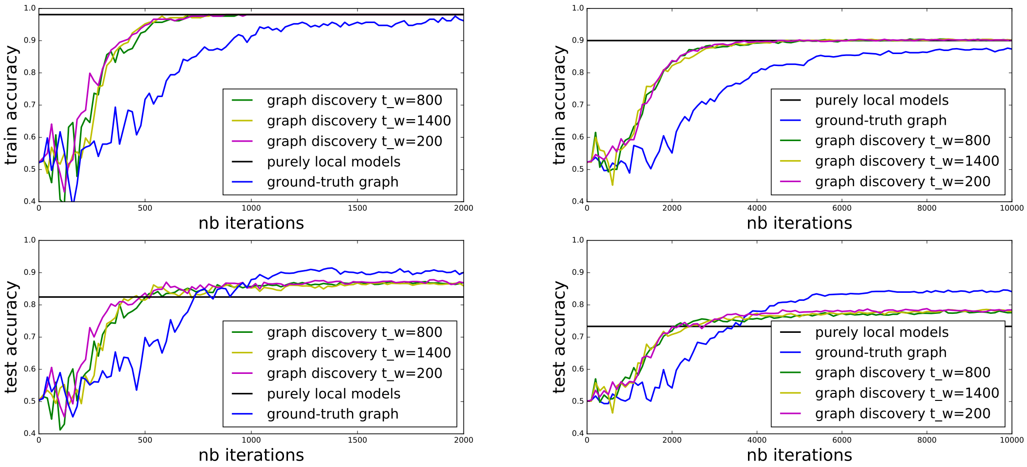
and test accuracy over the iterations. The results clearly show the gain in accuracy provided by our method. **Dada** is successful in reducing the overfitting of the purely local models, and yields higher test accuracy than both a global l_1-Adaboost model and personalized linear models.

4.6.2 Graph Discovery

In this second experiment, we learn the collaboration graph together with the models instead of relying on the ground-truth. Figures 4.5-4.6 show the effect of varying t_w (the number of iterations after which we update the graph) and q (number of neighbors). Overall, we see that our strategy to learn the collaboration graph can effectively make up for not knowing the ground-truth similarities. In particular, we are able to significantly outperform the purely local models in terms of test accuracy for all settings. Our approach does not seem to be very sensitive to t_w, which confirms that the graph can be updated fairly rarely. The neighborhood size q can be set to rather small values with negligible impact on the accuracy. This is important as the scalability of our decentralized algorithm heavily depends on the fact that each agent only needs to communicate with a small number of neighbors.

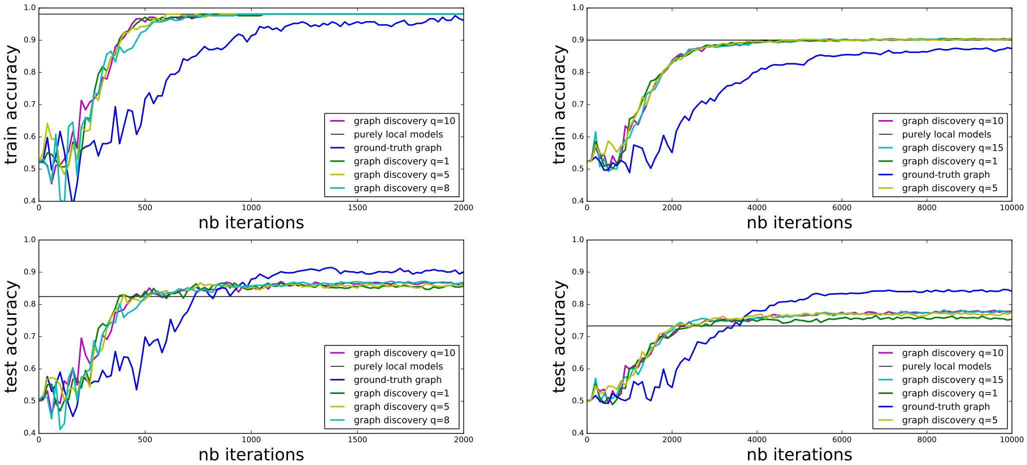
4.7 Conclusion

In this chapter, we illustrated a new decentralized technique for collaboratively learning personalized models over a graph of users which is both effective and communication efficient. We formulated the learning problem as an l_1-Adaboost



(a) $K = 20$, $q = 8$. (b) $K = 100$, $q = 15$.

Figure 4.5 – *Graph discovery evaluation with variable* t_w.



(a) $K = 20$, $t_w = 800$. (b) $K = 100$, $t_w = 800$.

Figure 4.6 – *Graph discovery evaluation with variable* q.

which jointly optimizes the overall empirical error and the smoothness between users' models, based on the graph topology. We made use of Frank-Wolfe technique for optimization for its intrinsic sparsity which makes it suitable for collaborative learning with low communication cost. As it might be hard to estimate the similarities between users a priori, we additionally proposed to simultaneously learn the local models and the graph topology through an alternating optimization procedure. The theoretical analysis proves the convergence in expectation of **Dada** and the empirical evaluation highlights how our method significantly reduces the overfitting of the purely local models, and improves the test accuracy w.r.t. both a global l_1-Adaboost model and personalized linear models.

We envision several promising research directions. To further enhance the applicability of the proposed framework, we would like to extend our approach to (functional) gradient boosting [71], where the set of base functions can be infinite. Frank-Wolfe algorithms can also be used in this setting [195], but how to efficiently implement graph regularization in infinite dimensions is an open question. Another promising avenue for future work is to make the proposed algorithm differentially-private [58]. As our algorithm communicates very little information, we believe that its privacy-accuracy trade-off may be better than the one of the method proposed in [14].

Part III

Learning using Landmark Similarities

Landmark SVM

5

This chapter is based on the publication

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. L ^{3}-svms: Landmark-based linear local support vector machines. In *CAp*, 2017.

Starting from this chapter, we make use of a set of points – namely landmarks – spread over the input space to capture its local characteristics. We will explicitly form a latent space by considering the similarities between the inputs and these landmarks. The similarities will be captured either by kernel functions or by linear approximations of them. Doing so, we drop the assumption that the peculiarities of the distribution are constant for a given subset of the partition.

Because of their appealing properties, such as scalability to large training sets, we still optimize linear models. However, instead of learning a local model per subset of data, we train a unique learner for the entire sample on a latent space overall suited to the task at hand. More precisely, we introduce a local adaptation to the well-known Support Vector Machines (SVMs) method, that we name **L^3-SVMs**. The main challenge will be taking advantage of the discriminatory power of kernel SVMs while making them scale to large datasets.

Simple and effective, our algorithm is also theoretically well-founded. Using the framework of Uniform Stability, we show that our SVM formulation comes with generalization guarantees on the true risk. The experiments based on the simplest configuration of our model (i.e. landmarks randomly selected from the training set, linear projection, linear kernel) show that **L^3-SVMs** is very competitive w.r.t. the state of the art and opens the door to new exciting lines of research.

This chapter is organized as follows: Section 5.1 gives an overview of the state of the art techniques for scaling SVMs to datasets large in number of samples and dimensionality; In Section 5.2 we present our method and analyze its computational and memory complexity; Furthermore, in Section 5.3 we study its theoretical guarantees using the framework of uniform stability; In Section 5.4 we finally carry out experiments first on synthetic datasets, in the aim of showing, and on real ones.

5.1 Locally linear SVMs

One of the most famous and commonly used Machine Learning techniques for classification are the Support Vector Machines (SVMs) [49]. This popularity is due to their robustness, simplicity, efficiency (even in non linear scenarios by means of the kernel trick) as well as their theoretical foundations via generalization guarantees.

Despite these nice properties, SVMs may face some drawbacks: as highlighted in Chapter 2, kernel SVMs are known to be expensive in terms of time complexity and memory usage when the number of training examples is large, both at training and at test time. For training, the full Gram matrix needs to be evaluated (i.e., compute and store all pairwise training sample similarities). For testing, the time complexity depends on the number of support vectors which typically grows linearly with the number of training instances [174]. Therefore, kernel SVMs have been shown not to scale well to very large data sets.

Over the years, several methods have been proposed to speed up SVMs, for instance by reducing the size of the training set [7], or by making use of stochastic optimization [27] or by solving an alternative formulation of the orginal SVM problem [90]. On the other hand, *locally-linear learning approaches* have been shown to be the most appealing in terms of training time, testing time and accuracy. As already mentioned in this manuscript, they are effective for data sets that present multi-modalities and/or non-linearities because they are able to capture the local characteristics of the space. They are also computationally efficient as they learn only linear classifiers (for which efficient solvers exist) and, at test time, are independent on the number of support vectors, because they give an explicit formulation of the decision function. However, these strategies suffer from the typical drawbacks of local learning approaches, such as over-fitting (see Chapter 2). We can distinguish between two main families of local SVM approaches: the ones that locally learn combinations of a set of learned linear SVMs as in [64, 107, 194], and those which partition the input space and learn a local model per region [73, 80, 198].

Methods from the first category estimate local combinations of linear SVMs and make the assumption that the input data is lying on a manifold along which the linear classifier evolves smoothly. In [107], the manifold is approximated by selecting some anchor points (using K-means) and learning one local model per anchor point. Each training point is then expressed, using a local coding scheme, as a linear combination of its closest anchor points. The local coding ensures that

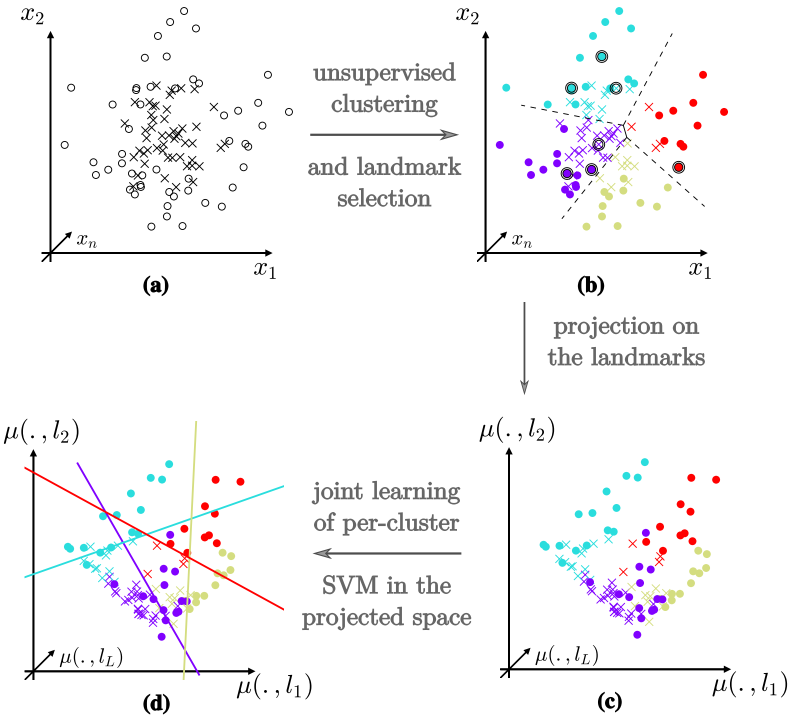


Figure 5.1 – *Illustration of the proposed approach.* ***L^3-SVMs*** *first partitions the input space into K clusters (represented with 4 colors in the figures) and randomly selects landmarks (7 circled points in (b)). All points are then projected on all landmarks (b$\mapsto$c) ($\mu(x, l_p)$ the projection of a point x on the p^{th} landmark). K locally linear models, each with its own support vectors but in a common projection space induced by all landmarks, are then learned by solving a joint problem (the separators are represented as colored lines in (d)).*

the prediction for each point is influenced by a limited number of models, thereby making the learning efficient. A latent SVM formulation is used in [64] where the authors follow the principle of [107] but extend it to a multi-class setting and replace the local coding scheme by latent coordinates that are estimated jointly with the parameters of the linear models.

Methods from the second family, such as clustered SVMs [80], first partition the input space, typically using K-means, and then learn a linear model in each region. A review of this family of approaches is reported in Chapter 2.

In this Chapter, we introduce a new local SVM method, called **L^3-SVMs**, that targets computational efficiency while having provable theoretical guarantees. Our method clusters the input space, carries out dimensionality reduction by projecting all points on selected landmarks, and learns interdependent linear combinations of linear models (see Figure 5.1 for an illustration).

As such, our method lies in between the two families of local approaches presented above without suffering from the aforementioned drawbacks. On one hand, the proposed method can be seen as learning a linear model on a latent space (linear or not) induced by the set of landmarks that is common to all clusters. On the other hand, it can be seen as clustering the input space and learning, in a projected space, a set of interacting linear models.

Using the framework of the Uniform Stability [30], we prove that our algorithm is stable w.r.t. changes in the training set allowing us to derive a tight generalization bound on the true risk. It is worth noticing that our algorithm, which can be interpreted as a generalization of the standard SVM formulation, is configurable and offers many points for improvement: clustering algorithm, regularization terms, landmark selection method, projection function, etc. While many variations can be imagined, our early experiments surprisingly show that the "default" choices (K-means clustering, random landmarks, linear projection) already yield an algorithm that is competitive with the state of the art while extremely fast and scalable.

5.2 Soft-margin Landmark-based Linear Local SVMs

In the next Section, we formally present **L^3-SVMs** and analyze its complexity in terms of memory and amount of computations. Our method consists in partitioning the input space into K clusters and learning K corresponding (linear) models that interact in a single optimization problem. The interactions come from a projection on a set of landmarks $\mathcal{L}$ that is common for all clusters and from the formulation of a unique linear problem with a single bias parameter. It is worth noting that a standard SVM is a particular case of our approach for $K = 1$ and specifically chosen landmarks.

5.2.1 Notations and Optimization Problem

Let $X \subseteq \mathbb{R}^d$ be the input space, $\mathcal{Y} = \{-1, 1\}$ the output space and $\{R_k\}_{k=1}^K$ a partition of $\mathcal{X}$ (learned, for instance, using K-Means). We consider a training sample $\mathcal{S} = \{z_i = (x_i, y_i, k_i)\}_{i=1}^m$ of m i.i.d. instances $z_i \in \mathcal{X} \times \mathcal{Y} \times \{1, .., K\}$ (such that $x_i \in R_{k_i}$) drawn from an unknown distribution $\mathcal{D}$. Moreover, we denote $\mathcal{L} = \{l_p\}_{p=1}^L \in \mathcal{X}^L$, a set of L landmarks of the input space (e.g. selected randomly from the training sample). The objective function of **L^3-SVMs** is defined as follows:

$$F(f) = \frac{1}{2} \|f\|^2 + \frac{c}{m} \sum_{i=1}^m \ell(f, z_i)$$

where $\ell(f, z) = \max(0, 1 - yf(x, k))$ is the hinge loss and $f : \mathcal{X} \times \{1, .., K\} \to \mathbb{R}$ is the function

$$f(x,k) = \sum_{p=1}^{L} \theta_{kp}\mu(x,l_p) + b$$

that is used for prediction with: $\hat{y} = sign(f(x,k))$.

Note that $\theta \in \mathbb{R}^{K\times L}$ is a matrix of weights expressing the influence of each landmark p for a given cluster R_k. Doing so, we are supposing that the problem is linear in the space created by both clusters and landmarks. Thus, we learn a vector of weights per cluster but a unique offset b.

Another way to see our method is as learning a unique SVM classifier in a projected space defined by the selected landmarks $\mathcal{L}$ and by a score function $\mu : \mathcal{X}^2 \to \mathbb{R}$ between points of the input space:

$$f(x,k) = \theta_{k.}\mu_{\mathcal{L}}(x)^T + b$$

where $\mu_{\mathcal{L}}(.) = [\mu(.,l_1),...,\mu(.,l_L)]$ is a projection from the input space $\mathcal{X}$ to the landmark space $\mathcal{H} \subset \mathbb{R}^L$. The score function μ could be, for instance, the scalar product $\mu(x_i,l_j) = x_i l_j^T$ or the RBF function $\mu(x_i,l_j) = \exp\left(-\frac{\|x_i - l_j\|_2}{2\sigma^2}\right)$. For instance, for μ the dot product, the clusters allow us to capture the non-linearities of the space while the landmarks help to control the size of the input space. Additionally, projecting on the landmarks acts as a regularization: as the landmarks are chosen without considering their class and the projection of an instance uses all the landmarks and not only those belonging to its partition, the risk of over-fitting is reduced. Therefore, unlike clustered SVMs [80], we don't need to learn an additional global model to regularize the local ones.

As previously mentioned, our method is a generalization of standard SVMs: it is similar to SVMs when $K = 1$ and the set of landmarks $\mathcal{L}$ forms a basis of the input space $\mathcal{X}$, and fully equivalent if this basis is also orthonormal.

A Soft-Margin version of our optimization problem can be written as follows:

$$\begin{gathered}\underset{\theta,b,\xi}{\arg\min} \frac{1}{2}\|\theta\|_{\mathcal{F}}^2 + \frac{c}{m}\sum_{i=1}^{m}\xi_i \\ s.t.\ \ y_i\left(\theta_{k_i.}\mu_{\mathcal{L}}(x_i)^T + b\right) \geq 1 - \xi_i \ \forall i = 1..m \\ \xi_i \geq 0 \ \forall i = 1..m\end{gathered}$$

which boils down to maximizing the margin between the class hyperplanes while minimizing the average classification error.

The previous problem is defined for the linear case but it can be easily rewritten for kernel SVMs considering that there exists an unknown mapping ϕ from $\mathcal{H}$ to $\mathcal{J}$, a space with potentially infinite dimensions, such that $\phi(\mu_{\mathcal{L}}(x_i))^T\phi(\mu_{\mathcal{L}}(x_j)) =$

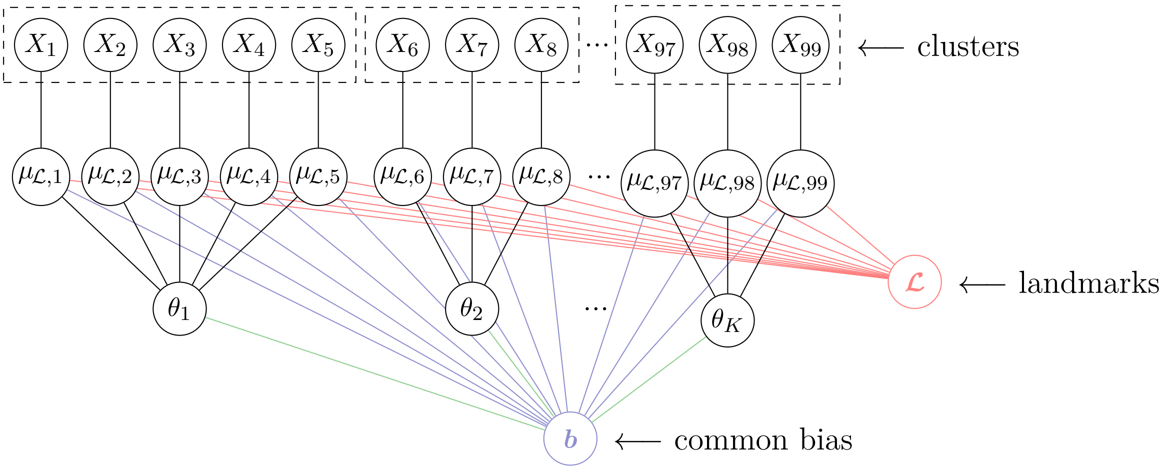


Figure 5.2 – *Variable dependencies for our model,* ***L***^{3}***-SVMs****, where one SVM is learned per cluster but the local models interact through a common bias and* $\mathcal{L}$*, the set of landmarks. A node represents a variable (or a set of) and a link show a direct dependency between the variables, i.e., one variable is directly involved in the computation or the estimation of the other.*

$kernel(\mu_{\mathcal{L}}(x_i), \mu_{\mathcal{L}}(x_j))$. The kernelized problem can be solved using its dual Lagrangian formulation (see Appendix E). Notice that the advantages of locally learning non-linear SVMs are limited, as our approach already captures non-linearity and has lower complexity compared to kernel SVMs. Moreover, the dual formulation would suffer from the same drawbacks of kernel SVMs. However, by solving the problem in its dual form, we can also study the relation between the learned model and the support vectors. The parameters are computed as follows (with $\{z_a = (x_a, y_a, k_a)\}_{i=1}^{A}$ the set of A support vectors and α_a the dual value of z_a):

$$\theta_{kp} = \sum_{a=1|k_a=k}^{A} \alpha_a y_a \mu(x_a, l_p)$$

$$b = \frac{1}{A}\sum_{a=1}^{A}(y_a - \theta_{k_a.}\mu_{\mathcal{L}}(x_a))$$

which means that the weight θ_{kp} for a cluster k and a landmark l_p depends on the support vectors of that particular cluster and on their similarities with l_p, while the parameter b is computed using the global information obtained from all the support vectors. Figure 5.2 gives a graphical illustration of the variable dependencies for **L**^{3}**-SVMs** (see Appendix E for a comparison with other local SVMs methods).

5.2.2 Computational Analysis

As previously mentioned, the main drawback of kernel SVMs is their inability to scale to large datasets. As a matter of fact, their training complexity is cubic with the number of instances and their testing and memory complexities depend on the number of support vectors which is $O(m)$ [174].

The proposed approach, if solved in its primal (e.g. using [60]), has a complexity close to linear SVMs while capturing non-linearities. In Table. 5.1, we compare **L^3-SVMs** with standard Linear-SVMs, Linear-SVMs applied on polynomial features of second order (Poly-SVM) and RBF-SVMs in terms of training, testing and memory (for storing the learned model) complexities. For **L^3-SVMs** we consider the default configuration (that is also used in the experiments of Sec. 5.4): clustering with K-means, random selection of landmarks and projection with the dot product.

The training complexity of our method could also be improved by using recent optimization techniques proved to reduce the training time, such as [7, 27].

Table 5.1 – *Computational comparison, with K: the number of clusters ($K \ll m$), L: the number of landmarks ($O(d)$), with d: the number of features, and m: the number of training instances.*

	Training Time	**Testing Time**	**Memory Usage**
Linear-SVM	$O(dm)$	$O(d)$	$O(d)$
Poly-SVM	$O(d^2m)$	$O(d^2)$	$O(d^2)$
RBF-SVM	$O(m^3)$	$O(dm)$	$O(dm)$
L^3-SVMs	$O(KLm + Ldm)$	$O(Ld)$	$O(KL + Ld)$

5.3 Theoretical results

In this section, we present a generalization bound on the true risk induced by our algorithm using the theoretical framework of the Uniform Stability [30] presented in Chapter 1. We will see that this theoretical analysis gives some insights about the number of landmarks to select in practice.

5.3.1 L^3-SVMs's Uniform Stability

We briefly recall the notion of Uniform Stability. The idea of Uniform Stability is to check if an algorithm produces similar solutions from datasets that are slightly different. Let S be the original dataset and S^i the set obtained after having replaced the i^{th} example of S by a new sample z_i' drawn according to $\mathcal{D}$. We will say that an algorithm is uniformly stable if the difference between the loss suffered (on a new instance) by the hypothesis f learned from S and the loss suffered by the hypothesis f^i learned from S^i converges in $O(\frac{1}{m})$.

For the following analysis, we introduce a new notation that allows us to simplify the derivations. We rewrite

$$f(x,k) = \theta\, \mu_{\mathcal{L}}(x)^T$$

with $\theta = [\theta_{0.}, ..., \theta_{k.}, ..., \theta_{K.}, b]$ and $\mu_{\mathcal{L}}(x) = [\mathbf{0}, ..., \mu_{\mathcal{L}}(x), \mathbf{0}, ..., \mathbf{0}, 1]$ (that implicitly depends on k) both of size $KL+1$ and

$$F(f) = \frac{1}{2}\left\|\theta\right\|^2 + \frac{c}{m}\sum_{i=1}^{m} \max(0, 1 - y_i(\theta\mu_{\mathcal{L}}(x_i)^T)) \tag{5.1}$$

To derive the theoretical guarantees, we make use of the property of σ-admissibility enjoyed by the hinge loss. In order for the algorithm to be stable, it is necessary to prove that, for a given point, the difference between its loss function evaluated for any two possible hypotheses is bounded by the difference of hypotheses' predictions, scaled by a constant. Following [30], we know that the hinge loss is 1-admissible.

We can now present the main result about our algorithm **L^3-SVMs**.

Theorem 5.1 **L^3-SVMs Uniform Stability** *Assuming that $\forall x \in \mathcal{X}, \|x\| \le c$, **L^3-SVMs** has uniform stability $\frac{cLM^2}{m}$, where $M = \max(c^2, 1)$ if μ is the dot product and $M = 1$ if μ uses the RBF kernel.*

Proof. As $\ell(f,z)$ is 1-admissible, $\forall z = (x,y,k) \in \mathcal{Z}$,

$$\left|\ell(f^{\backslash i}, z) - \ell(f,z)\right| \le \left|f^{\backslash i}(x,k) - f(x,k)\right| = |\Delta f(x,k)| \tag{5.2}$$

with $\Delta f = f^{\backslash i} - f$. By denoting $\Delta\theta = \theta^{\backslash i} - \theta$, we can derive, $\forall z = (x,y,k) \in \mathcal{Z}$,

$$\begin{aligned}
|\Delta f(x,k)| &= \left|\theta^{\backslash i}\mu_{\mathcal{L}}(x)^T - \theta\mu_{\mathcal{L}}(x)^T\right| \\
&= \left|(\theta^{\backslash i} - \theta)\mu_{\mathcal{L}}(x)^T\right| \\
&\le \left\|\theta^{\backslash i} - \theta\right\|_{\mathcal{F}} \left\|\mu_{\mathcal{L}}(x)\right\| \qquad (5.3)\\
&\le \left\|\Delta\theta\right\|_{\mathcal{F}} \left\|\mu_{\mathcal{L}}(x)\right\| \\
&\le \left\|\Delta\theta\right\|_{\mathcal{F}} \sqrt{L} \left\|\mu_{\mathcal{L}}(x)\right\|_\infty \qquad (5.4)\\
&\le \left\|\Delta\theta\right\|_{\mathcal{F}} \sqrt{L} \max_l(\mu(x,l)) \\
&\le \left\|\Delta\theta\right\|_{\mathcal{F}} \sqrt{L}M. \qquad (5.5)
\end{aligned}$$

Eq. (6.4) is due to the Cauchy-Swartz inequality, and Eq. (6.5) is because $\|\mu_{\mathcal{L}}(x)\| \le \sqrt{L}\,\|\mu_{\mathcal{L}}(x)\|_\infty$ recalling that $\mu_{\mathcal{L}}(x) \in \mathbb{R}^{(1\times L)}$.

The value of M depends on the chosen function μ. For instance, if μ is the dot product, $M = \max(C^2, 1)$ and if it uses the RBF kernel, $M = 1$.

From Lemma 21 of [30]:

$$2 \left\|\Delta\theta\right\|_{\mathcal{F}}^2 \leq \frac{c}{m} \left|\Delta f(x_i, k_i)\right| .$$

Then, by instantiating Eq. (6.6) with $z = z_i$, we get

$$\left\|\Delta\theta\right\|_{\mathcal{F}}^2 \leq \frac{c}{2m} \left|\Delta f(x_i, k_i)\right| \leq \frac{c}{2m} \left\|\Delta\theta\right\|_{\mathcal{F}} \sqrt{L}M$$

and as $\left\|\Delta\theta\right\|_{\mathcal{F}} > 0$, we obtain

$$\left\|\Delta\theta\right\|_{\mathcal{F}} \leq \frac{c}{2m}\sqrt{L}M.$$

So, from the previous bound on $|\Delta f(x, k)|$, we get

$$\forall z = (x, y, k), \quad \left|\Delta f(x, k)\right| \leq \left\|\Delta\theta\right\|_{\mathcal{F}} \sqrt{L}M \leq \frac{cLM^2}{2m}$$

which, with Eq. (6.3) gives the $\frac{cLM^2}{m}$ uniform stability.

□

Note that the stability of the algorithm depends on the number of selected landmarks. **L^3-SVMs** is stable only if $L \ll m$, which is not a strict condition considering that, in practice, we select $L = O(n)$ landmarks (with n the size of the input space $\mathcal{X}$) and that, for learning in general, $n \ll m$. The choice of the number of landmarks is, then, crucial. The landmarks needed for good predictions depends on the distribution of the data. As a matter of fact, the diversity of the landmark set allows to capture the modalities of the input data, so that the more complex is the distribution of the data, the more diversified landmarks are required. On one hand, a limited number of landmarks will be probably unable to represent the input data, especially if they are chosen randomly. On the other hand, when the number of landmarks is too high, the model is more likely to over-fit, as expressed by the generalization bound that we derived, because the landmark set will probably adapt too much to the training distribution.

5.3.2 Generalization bound

For the ease of reading, we report the formulation of the generic bound derived using the framework of Uniform Stability.

Theorem 5.2 *[30] Let A be an algorithm with uniform stability $\frac{2\beta}{m}$ w.r.t. a loss ℓ such that $0 \leq \ell(f,z) \leq E$, $\forall z \in \mathcal{Z}$. Then, for any i.i.d. sample S of size m and for any $\delta \in (0,1)$, with probability $1-\delta$:*

$$R_{\mathcal{D}}(f) \leq \hat{R}_S(f) + \frac{2\beta}{m} + \left(4\beta + E\right)\sqrt{\frac{\ln\frac{1}{\delta}}{2m}}$$

where $R_{\mathcal{D}}(f)$ is the true risk and $\hat{R}_S(f)$ is the empirical risk on sample S.

Corollary 5.1 *The generalization bound of **L^3-SVMs** derived using the Uniform Stability framework is as follows:*

$$R_{\mathcal{D}}(f) \leq \hat{R}_S(f) + \frac{cLM^2}{m} + \left(\frac{2cLM^2}{m} + 1 + 2c\sqrt{L}M\right)\sqrt{\frac{\ln\frac{1}{\delta}}{2m}}.$$

Proof. For deriving the generalization bound, we need to prove that our loss ℓ is bounded by a constant E when evaluated at the optimal solution of F (5.1). Let f be the minimizer of F. We deduce that:

$$\begin{gathered} F(f) \leq F(\mathbf{0}) \\ \frac{1}{2}\left\|\theta\right\|^2 + \frac{c}{m}\sum_{i=1}^{m}\max(0, 1 - y_i(\theta\mu_{\mathcal{L}}(x_i)^T)) \leq \frac{1}{2}\left\|\mathbf{0}\right\|^2 + \frac{c}{m}\sum_{i=1}^{m}\max(0, 1 - y_i(\mathbf{0}\mu_{\mathcal{L}}(x_i)^T)) \\ \frac{1}{2}\left\|\theta\right\|^2 \leq c \qquad (5.6) \\ \left\|\theta\right\|^2 \leq 2c \end{gathered}$$

Eq. (6.8) is because $\forall a, b, c \in \mathbb{R}^+$, $a + b \leq c$ implies that $b \leq c$.

Thus,

$$\begin{aligned} \ell(f,z) &= \max(0, 1 - y\theta\mu_{\mathcal{L}}(x)^T) \\ &\leq 1 + \left|\theta\mu_{\mathcal{L}}(x)^T\right| \\ &\leq 1 + \left\|\theta\right\|\left\|\mu_{\mathcal{L}}(x)^T\right\| \qquad (5.7) \\ &\leq 1 + 2c\sqrt{L}M = E \end{aligned}$$

Eq. (6.9) comes again from the Cauchy-Swartz inequality.

□

5.4 Experimental results

In this section, we empirically study the behavior of **L^3-SVMs** both on synthetic and on real datasets, and for binary and multiclass classification. Specifically, we study the impact of the number of clusters and the number of landmarks on learning, we analyze two different methods for selecting the landmarks and finally we compare our method to the state-of-the-art SVM based techniques. **L^3-SVMs** is implemented in Python using the liblinear [60] library and the multiclass classification is performed through a one-vs-all procedure. The code is available at `https://github.com/vzantedeschi/L3SVMs`.

5.4.1 Capturing the Non-Linearities

Here we study the influence of the number of clusters on learning. We compare the performances of standard SVMs (linear or kernelized with a RBF kernel) with those of **L^3-SVMs** (using the inner product or the RBF projection function) on two toy non-linearly separable distributions: the XOR distribution and the Swiss-roll distribution. Remember that, for our method, even if the function μ used for projecting the data is the RBF, the learned models are still linear and the learning remains efficient.

For these experiments, we tune the hyper-parameters of each method by grid search with the values $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$ in a 5-fold cross-validation procedure and for the **L^3-SVMs**, the number of landmarks is arbitrarily fixed to 10 which are randomly selected from the training sample. The instances are clustered using K-means. In Fig. 5.3 and 5.4, we draw the learned class separators, as well as the training instances (according to their true label) and the support vectors marked by a black dot. We report the training and test accuracies (on training and test samples of same size) and the number of support vectors.

XOR distribution, Fig. 5.3 We generated a synthetic XOR distribution by drawing instances uniformly over a 2D-space and assigning to each instance the label $+1$ (resp. -1) if its coordinates have the same sign (resp. different signs). As expected, the linear SVM is not able to separate the two classes, while the RBF SVM captures the non-linearities of the space. We notice that the performances of a **L^3-SVMs** are comparable to the RBF SVM in terms of accuracy and number of support vectors even with just 2 clusters and that with 4 clusters we achieve the best results. Moreover the learned class regions are similar to the theoretical ones.

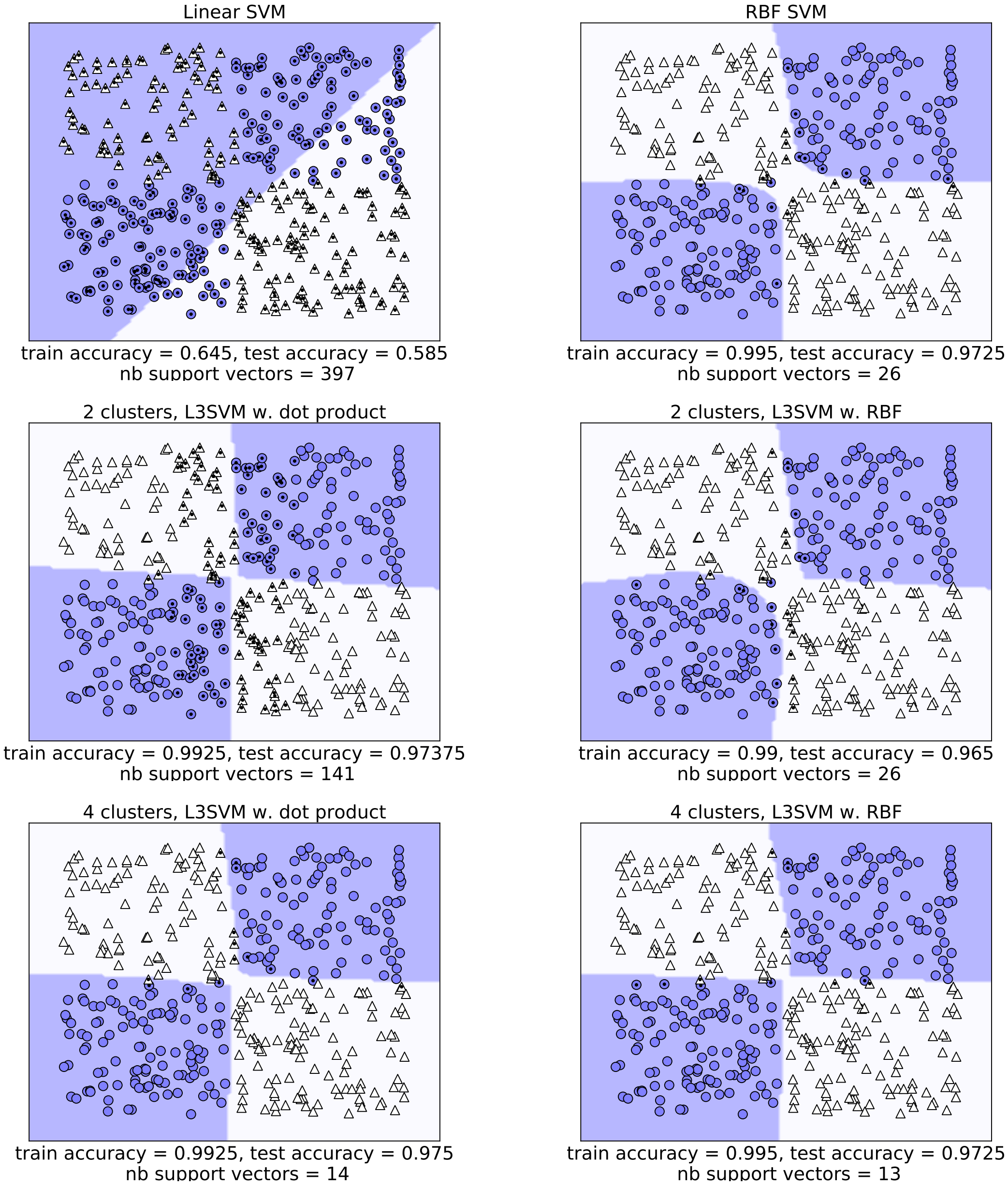


Figure 5.3 – *2D-XOR distribution: 400 training instances. The two colors depict the regions of the predicted classes, while the training instances are drawn according to their true label using two different shapes. The support vectors marked by a black dot.*

Swiss-roll distribution, Fig. 5.4 The problem consists in separating a Swiss-roll distribution (the first class) from a uniform one (the second class). Unlike the XOR distribution, in this case 2 clusters are not enough to capture the non-linearities of the space, but with 100 clusters we obtain better performance than the ones of kernel SVMs.

Notice that, in both experiments, as the number of clusters increases, the difference in accuracy between a **L^3-SVMs** with a very fast inner product and a **L^3-SVMs** with a RBF projection function is irrelevant. Our method is then able to capture the non-linearities of the space as well as a non-linear SVM. Note that the number of clusters depends, above all, on the nature of the input space.

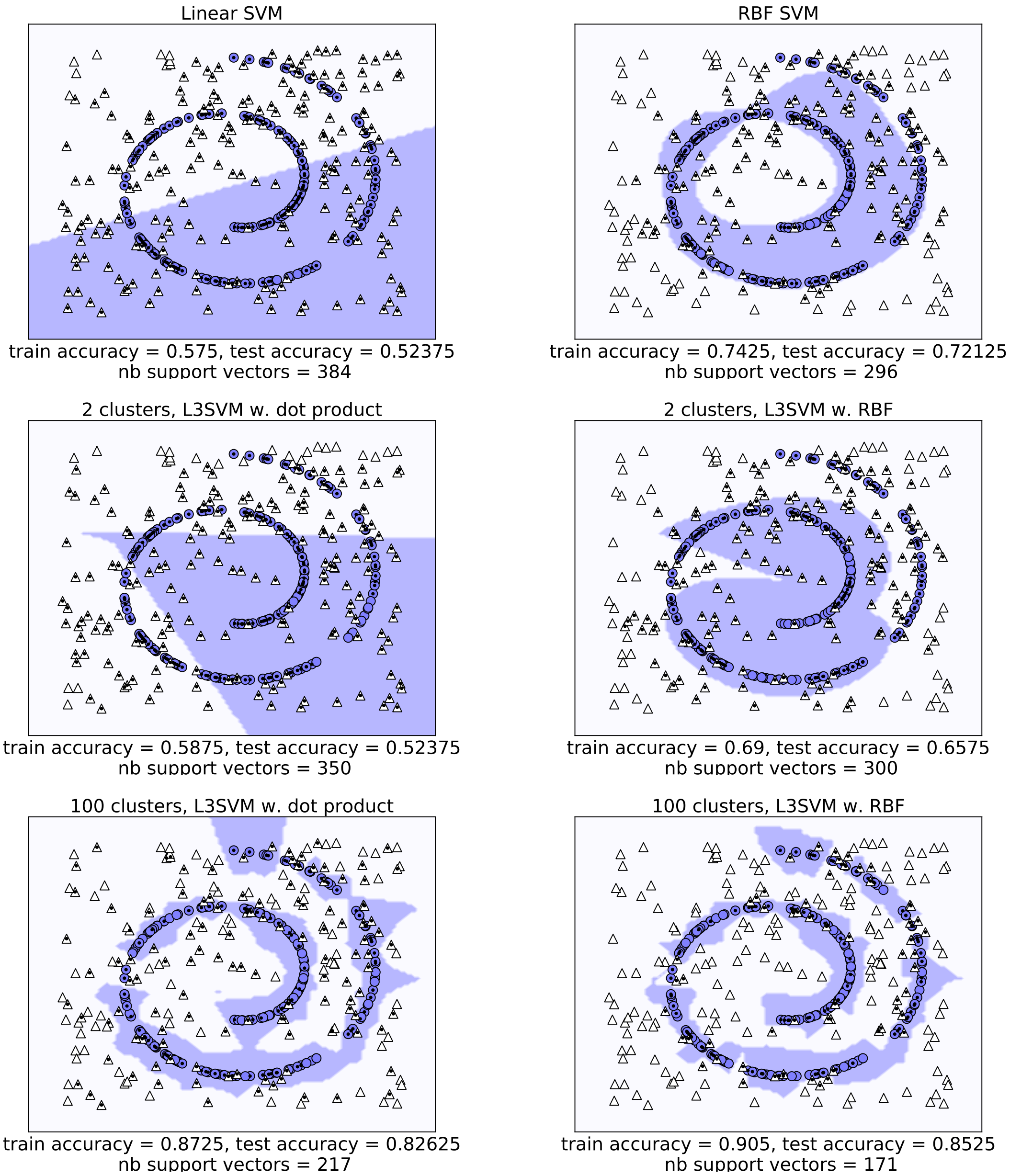
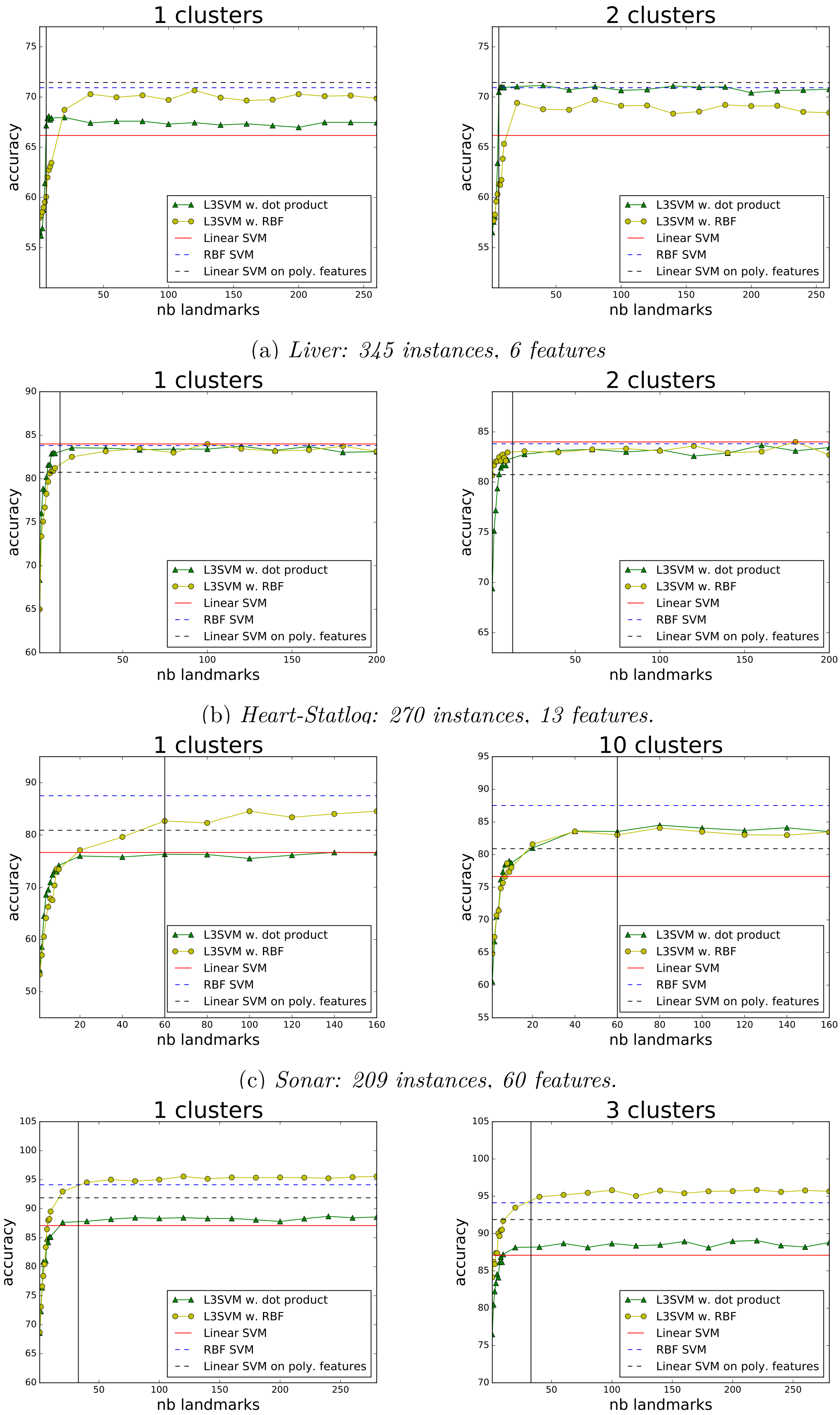


Figure 5.4 – *2D-Swiss-roll distribution: 400 training instances, balanced classes. The two colors depict the regions of the predicted classes, while the training instances are drawn according to their true label using two different shapes. The support vectors marked by a black dot.*

5.4.2 Fixing the number of landmarks

The aim of the following experiment is to empirically study how the number of landmarks impacts the test accuracy. To do so, we fix the number of clusters



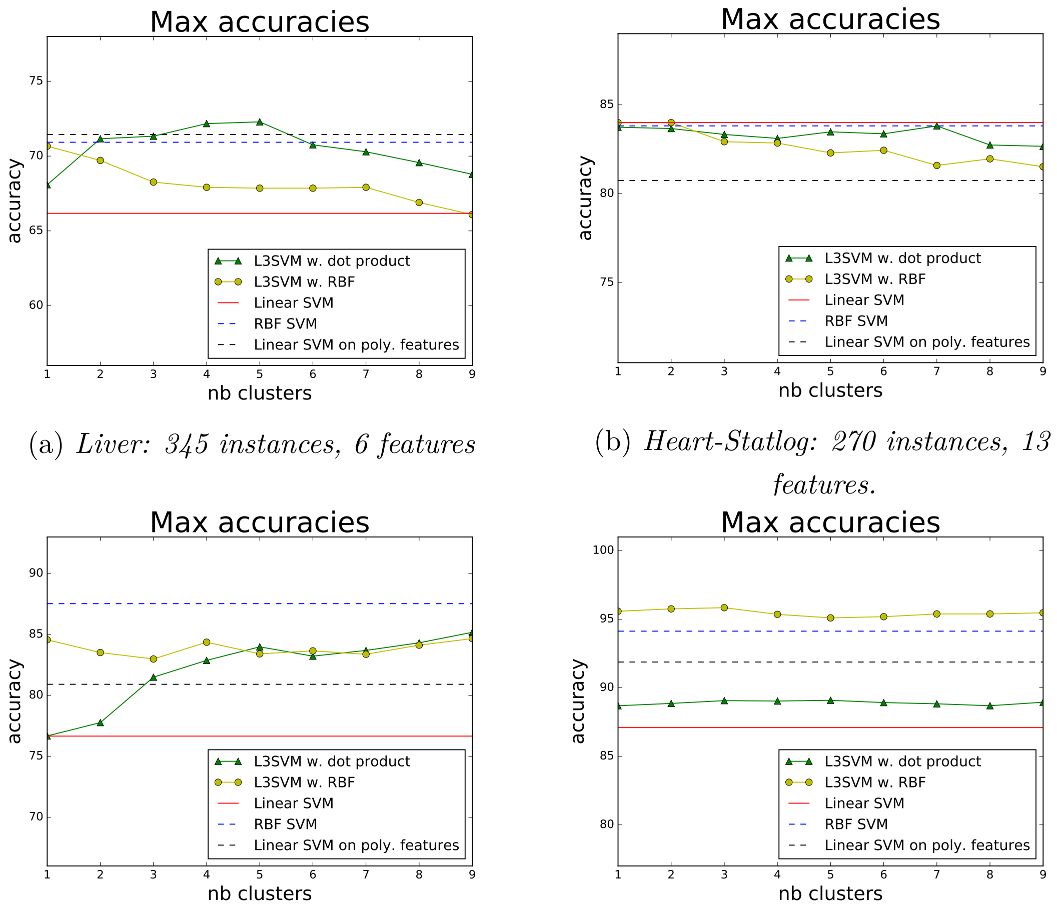
(a) *Liver: 345 instances, 6 features*

(b) *Heart-Statlog: 270 instances, 13 features.*

(c) *Sonar: 209 instances, 60 features.*

(d) *Ionosphere: 351 instances, 33 features.*

Figure 5.5 – *We report the results for a fixed number of clusters and increasing number of landmarks. The black line in the pictures marks the size of the dataset.*



(a) *Liver: 345 instances, 6 features*

(b) *Heart-Statlog: 270 instances, 13 features.*

(c) *Sonar: 209 instances, 60 features.*

(d) *Ionosphere: 351 instances, 33 features.*

Figure 5.6 – *For 1 to 9 clusters we report the maximal mean accuracy obtained with all possible values of L with a given number of clusters.*

(between 1 and 9) and vary the number of landmarks from 1 to the size of the training sample.

We compare the performances of standard SVMs (linear, linear on polynomial features of second order or kernelized with RBF) with those of **L^3-SVMs** (using a linear or RBF projection) on three UCI datasets [120]. In Fig. 5.5 we draw the mean testing accuracies of a 5-fold cross-validation procedure repeated 10 times. For all the methods, at each iteration we tune the hyper-parameters by grid search with the values $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$ with a 5-fold cross-validation procedure and we cluster the instances using k-means.

Liver, Fig. 5.5a and 5.6a Already with 2 clusters, **L^3-SVMs** achieves testing accuracies similar to those of a kernelized SVM. Furthermore, our method has the best results for 2 to 6 clusters. On the other hand, it seems that a **L^3-SVMs** with a RBF projection function is quite sensitive to overfitting. It is interesting to notice that, by fixing the number of clusters to 1, projecting on enough landmarks allows us to obtain a latent space more suited to the task.

Heart-Statlog, Fig. 5.5b and 5.6b In this case, learning local models makes the predictions worse than learning a global one. As a matter of fact, from the comparison of an SVM and a Kernel SVM, it seems that the problem is linearly separable and that learning a non-linear classifier does not improve the results. Therefore, increasing the number of local models only makes them overfit.

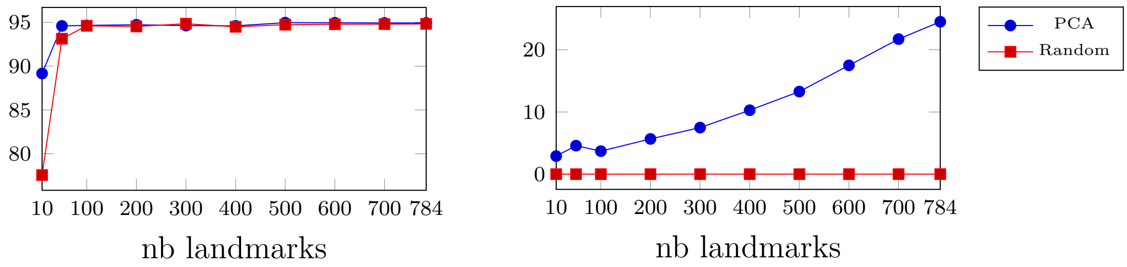
Sonar, Fig. 5.5c and 5.6c Studying this dataset, which has more features than the previous two, it seems that it is possible to select a number of landmarks smaller than the dimensionality of the input space without deteriorating the results.

Ionosphere, Fig. 5.5d and 5.6d Unlike all previous datasets, the number of clusters, in this task, seems to only slightly impact the performance of the final model. Consequently, **L^3-SVMs** with the dot product is incapable to attain the results achieved by the RBF-SVM.

In conclusion, we claim that it is not interesting to have a number of landmarks greater than the dimensionality of the input space and that reducing the number of landmarks is not conceivable on datasets of small number of features. Moreover, from Figure 5.6 we can evince that, when the number of landmarks is optimal, the accuracy of **L^3-SVMs** with a RBF projection is not improved by increasing the number of clusters. On the contrary, most of the times the accuracy is degraded by it. Finally, the performance of **L^3-SVMs** with a RBF projection function, in most cases, are close or even worse than those of **L^3-SVMs** with a linear kernel, probably because of overfitting. Therefore, in the following sections, we will restrict our studies only to **L^3-SVMs** with linear projection.

5.4.3 Dimensionality Reduction

The aim of this series of experiments is to study the impact of the chosen technique for landmark selection on the performances of our method. We compare **L^3-SVMs** with a set of landmarks randomly selected from the training sample to **L^3-SVMs** with the landmarks as the principal components of the covariance matrix of the training set (performing a PCA) on the MNIST dataset [115]. In Fig. 5.7, we report the testing accuracies w.r.t. the number of landmarks L, as well as the time needed for selecting the landmarks. The number of clusters is fixed to 100 and the



(a) *Testing Accuracy (%)* (b) *Selection Time (s)*

Figure 5.7 – *Comparison of the testing accuracies and selection times (in seconds) for two methods of landmark selections: PCA and random selection. Notice that the difference in accuracy is limited when L is bigger than* 100, *while the time complexity is significantly lower using a random selection (around* 0.020*s).*

Table 5.2 – *Characteristics of Datasets*

	#training	#testing	#features	#classes	#models
SVMGUIDE1	3089	4000	4	2	100
IJCNN1	49990	91701	22	2	100
USPS	7291	2007	256	10	80
MNIST	60000	10000	784	10	90

parameter c is tuned by grid search by 5-fold cross-validation. The instances are clustered using k-means.

We use the Principal Component Analysis of the scikit-learn package [155], which implements the randomized SVD presented in [81]. Having denoted the number of features by d and the number of instances by m, the complexity of this method is at worst $O(md\log(d) + (m+d)d^2)$, when the rank of the training set is equal to d. Compared to a random selection ($O(L)$ as $L \ll m$), the PCA-based selection is more expensive and it achieves better results only when $L < 100$.

These results suggest that, when L is small, it is interesting to select good landmarks (by means of a PCA for instance) and it can be done in reasonable time. On the other hand, when L is big enough, there is no need to force the variety and expressiveness of the set of landmarks, and a random selection from the training sample already allows us to have a good projection of the input space with little effort.

5.4.4 Comparison with the State of the Art

In this final series of experiments, we compare **L^3-SVMs** with state-of-the-art methods on the four datasets presented in Table 5.2 (with the features rescaled to have a standard deviation of 1). In all experiments, we fix L to the dimension of the input space, we select the landmarks randomly from the training sample and we cluster using k-Means.

Table 5.3 – *Testing Accuracies on real datasets(%)*

	SVMGUIDE1	**IJCNN1**	**USPS**	**MNIST**
RBF-SVM	96.53	97.08	94.07	96.62
Linear-SVM	95.38	89.68	91.72	91.8
CSVM	95.05	96.35	N/A	N/A
LLSVM	94.08	92.93	75.69	88.65
ML3	96.68	97.73	93.22	97.04
L^3-SVMs	95.73	95.74	92.12	95.05

Table 5.4 – *Testing Accuracies (%) and Training Speedups w.r.t. RBF-SVM.*

	SVMGUIDE1		**IJCNN1**		**USPS**		**MNIST**	
RBF-SVM	96.53	*1*x	97.08	*1*x	94.07	*1*x	96.62	*1*x
Poly-SVM	96.35	*2.1*x	92.65	*5.2*x	N/A	*N/A*	N/A	*N/A*
Linear-SVM	95.38	*9.8*x	89.68	*140.5*x	91.72	*30.6*x	91.8	*112.5*x
CSVM	95.05	*0.3*x	96.35	*45.2*x	N/A	*N/A*	N/A	*N/A*
LLSVM	94.08	*1.7*x	92.93	*16.8*x	75.69	*0.4*x	88.65	*1.9*x
ML3	96.68	*0.3*x	97.73	*5.9*x	93.22	*1.1*x	97.04	*2.1*x
L^3-SVMs	95.73	*1.8*x	95.74	*7.4*x	92.12	*1.3*x	95.05	*9.8*x

Table 5.4 reports the accuracy and running times for several methods: standard SVMs using either a linear kernel, a second-order polynomial kernel or an RBF kernel (using Liblinear or Libsvm [40]); Clustered SVM (CSVM) [80]; Locally Linear SVM (LLSVM) [107]; ML3 SVM [64] and **L^3-SVMs**. [1] The number of local models is fixed and, if not differently specified in the respective papers (such as 8 nearest neighbors for LLSVM and $p = 1.5$ for ML3), the hyper-parameters are tuned by 5-fold cross-validation.

We can note that for datasets with a large number of features (i.e. USPS, MNIST) **L^3-SVMs** is always more efficient in terms of training time than other methods while keeping good accuracies. This is mainly due to the fact that, in **L^3-SVMs**, the latent space is induced by a simple linear projection on randomly selected landmarks, while the other methods are often based on more sophisticated latent spaces. When the number of features is rather small, all methods provide both good accuracies and training time, except for LLSVM (especially on IJCNN1). Even though the testing time is sometimes higher than the other methods, it can be reduced by limiting the number of landmarks for the datasets with a lot of features, as in the previous experiments we showed that, up to a limit, it does not affect the results.

[1] The results of CSVM for the multi-class datasets are missing because it is implemented only for binary classification.

5.5 Conclusion

In this chapter, we introduced a new local learning algorithm named **L^3-SVMs**. It relies on a partitioning of the input space and on a projection of all points onto a set of landmarks. Using the Uniform Stability framework, we showed that **L^3-SVMs** has theoretically generalization guarantees. The empirical evaluation highlights that **L^3-SVMs** is fast while being competitive with the state of the art. Moreover, several experiments allowed us to gain insights into the relevancy of the elements of the method, notably the type and number of landmarks, the number of clusters and the choice of the kernel. In particular, we reached several conclusions:

1. the number of landmarks L can be upper bounded by the dimensionality d of the original input space;
2. when the number of landmarks is consequent, the set $\mathcal{L}$ can be selected randomly from the training sample without loss of accuracy;
3. partitioning the inputs into clusters is necessary while using linear projections, such as the dot product, but it deteriorates performance while using non-linear kernels, such as the RBF.

However, several avenues of research need to be explored yet. First, we can refine many of the elements of **L^3-SVMs**: the partitioning using K-means can be replaced by other existing hard or soft clustering algorithms; the random landmark selection procedure could be optimized, for example using methods like DSELECT [93] and Stochastic Neighbor Compression [104], or using density estimation [123]. As mentioned earlier, the method can also be sped up by using recent optimization techniques proved to reduce the training complexity, such as [7, 27].

Even though the common landmarks act as a regularization of the local models, in the experiments, an over-fitting phenomenon is observed when the number of clusters is very large. The model could naturally accept explicit spatial regularization terms to increase the spatial smoothness of the models across clusters, as done for **C2LM** in Chapter 3.

Finally, the speed and linearity of **L^3-SVMs** open the door to exciting perspectives, such as an auto-context approach (stacking of projections): **L^3-SVMs** could be reapplied on the data after projecting it on the support vectors of the previous level, and so on. Beyond stacking, work on a deep version of the algorithm is in progress, where we learn in a joint optimization problem the intermediate layers of projection.

In the next chapter, we will present an adaptation of **L^3-SVMs** to multi-view data, in which each instance is observed in multiple feature spaces, potentially heterogeneous. Due to the increased complexity of the data representation, in this first work we will not partition the data ($K = 1$) and we will rely only on non-linear kernels as projection functions in order to capture the idiosyncrasies of the input spaces. We will show how this new version of **L^3-SVMs** allows us to work in a unified space for all views while keeping a low computational complexity.

Landmark SVM for Multi-View Data 6

This chapter is based on the publication

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. Fast and provably effective multi-view classification with landmark-based svm. In *ECML PKDD*, 2018.

In this last chapter of contributions, we introduce a fast and theoretically founded method for learning landmark-based SVMs (**L^3-SVMs**) in a multi-view classification setting. In such scenario, the instances of the dataset are observed in multiple feature spaces. We argue that the key to effectively tackling multi-view problems is exploiting the diversity between views, as the different views rarely contain, alone, sufficient information for the task at hand. The proposed method – named **MVL-SVM** – leverages the complementary information of the different sources of information and linearly scales with the size of the dataset.

The approach applies a non-linear projection to the dataset through multi-view similarity estimates w.r.t. a set of selected landmarks, before learning a linear SVM in the latent space joining all the views. In this new setting, both instances of the sample and landmarks are observed in multiple feature spaces. We account for this multifaceted representation in the similarities between points and landmarks: a kernel needs to be selected per view and the similarities need to be computed one view at a time (to compare a point to a landmark, the features on a view are compared to the features of the landmark on the same view, so on and so forth for all the views.). In the final step, by concatenating the obtained view-wise representations of the sample, we manage to get a unique latent space, common to all the views.

In the light of the empirical results of Chapter 5, the set of landmarks is selected randomly from the training sample. Moreover, despite the effectiveness of coupling the dot product with the data partitioning for capturing the data distribution, and in the aim of simplifying the formulation of the problem, which has been overloaded by the increased complexity of the data representation, in the following work we only rely on the choice of kernel for representing the characteristics of the space and we do not perform any clustering of the data.

We prove the generalization capabilities of our algorithm using the the uniform stability framework: we first analyze the robustness of the approach to slight changes in the training set and, then, derive a tight generalization bound dependent on the number of views and landmarks. In a second moment, we extend our method to the missing-view scenario. Thanks to the formulation of our problem, we show that we can reconstruct the missing views of the incomplete points simply by estimating the similarities to the landmarks in the single views without approximating the missing feature values. Empirical results, both in complete and missing view settings, highlight the superior performances of our method, in terms of accuracy and execution time, w.r.t. state of the art techniques.

The Chapter is structured as follows: In Section 6.1, we present the multi-view setting, with its specificities and new challenges, and give a quick overview of the solutions of the state of art; In Section 6.2, we present **MVL-SVM** and, in Section 6.3, we derive its generalization guarantees in the form of an upper bound on the true risk; In Section 6.4, we detail an extension of our approach to the missing-view scenario; Finally, in Section 6.5, we carry out an empirical evaluation of **MVL-SVM**, both in complete and missing settings.

6.1 Multi-view learning

Machine learning has mainly focused, during the past decades, on settings where training data is embedded in a single feature set. However, data collected nowadays is rarely of a single nature. It is rather observed in multiple, possibly heterogeneous views, where each view can take the form of a different source of information. Examples of multi-view datasets are documents translated in different languages, corpora of pictures with descriptive captions, clips with both audio and video streams, or simply objects observed with different visual conditions (e.g. viewpoint and exposure) like in the recent works [42, 43].

Definition 6.1 *(Multi-view data) An instance x is said to be multi-view if it lies in multi-view space of V views $\{W_i \subseteq \mathbb{R}_i^d\}_{i=1}^V$, so that $x \in \mathcal{X} \subseteq \mathbb{R}^{d_1+\cdots+d_V}$.*

Dealing with such scenarios led to the development of the multi-view learning setting [176, 208, 227] facing new challenges and requiring scientific breakthroughs. Consider, for instance, the problem of detecting a neurological disorder from a series of brain images from which different brain parcellation atlases are extracted[1]. Each atlas (or view) corresponds to a different parcellation of the 3D representation of the brain (a functional MRI): regions of interest (ROIs) are highlighted on the scanning that vary from a view to another. In neuroimaging data analysis, the co-activations between these ROIs are extremely useful for detecting behavioral or clinical patterns. Figure 6.1 gives an example of two atlases, which are represented each by three 2D sections but are originally 3D. As the selected ROIs vary from atlas to atlas, each parcellation captures different information on the brain activity and it would be interesting to leverage the complementary information coming from all atlases. However, typical machine learning algorithms hinder exploitation of different sources of information.

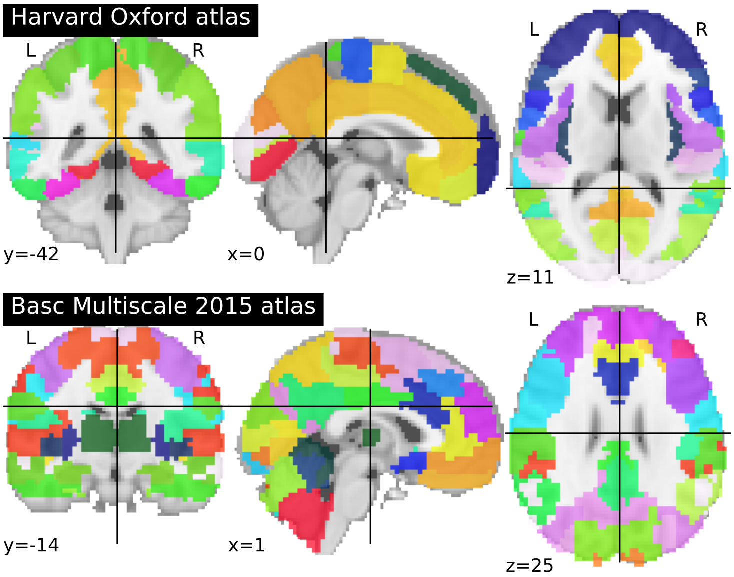


Figure 6.1 – *This figure shows two different views (here, atlases) of the brain at resting state. Each view shows in color the detected regions of interest. (Atlases available in the nilearn library* `http://nilearn.github.io/`*.)*

Basically, the need for designing multi-view algorithms relies on the observation that standard learning methods with good performance on single-view problems are, in most cases, inefficient in a multi-view setting [61, 88, 182]. Indeed, the views of an instance don't necessarily stand-alone because they might individually carry insufficient information about the task at hand. Even worse, they can be noisy or missing for a part of the training set. Thus, learning a model jointly on the ensemble of views has been proved to be more expressive than view-specific models, because it exploits the possible complementarity between views [227].

A rich literature of methods has been proposed over the years to provide solutions for extracting information from multiple sources. We can distinguish two principal

[1] A challenge has been organized in 2018 on the subject `https://paris-saclay-cds.github.io/autism_challenge/`

families of approaches which address multi-view problems: those which optimize a set of single-view learners and combine their predictions, and those which learn a single model in a common space shared by all views. We now provide an overview of the state of the art techniques for multi-view learning.

6.1.1 Multi-view Learning Algorithms

The simplest solution to tackle multi-view problems consists in working on the concatenated space of views, i.e. treating each view as a subset of features. However, as the nature of the views can be heterogeneous, i.e. their corresponding features might lie in different input spaces, such a solution is often unfeasible. Moreover, it does not take into account the statistical specificities of each view and can suffer from the curse of dimensionality.

Common multi-view state of the art approaches learn a set of single-view models either by *co-training* [26], in an attempt to capture both the commonalities and idiosyncrasies of the views, or by *co-regularization* [61, 175] over the predictions, aimed at maximizing their agreement (see [176, 208, 227] for surveys). Basically, these techniques train multiple view-specific models either by alternatively optimizing them, "teaching" one another, or by fostering their smoothness in predictions. The final step of such techniques consists in aggregating the predictions of the view-specific classifiers, for instance by majority vote [61, 175] or by weighted majority vote [78, 135]. Note that these methods usually face the following issues: their performances are degraded by the computational overload of training and testing multiple learners; also, by usually making the assumption that the views' common information is the only one worth keeping, they boil down to denoising the single views from their uncorrelated information. Yet, it is worth noticing that the information relevant to the task is not necessarily the one the views share, but the one that can be extracted by aggregating the views' incomplete information.

A few techniques [88, 92, 134] have also been proposed suggesting to address the problem in a unified space common to all views, allowing us to learn a single model while exploiting the different sources of information. These methods utilize a Vector-valued Kernel Hilbert Spaces (vvRKHS) [133], whose reproducing kernel outputs, for a pair of multi-view points, a matrix of similarities, each component weighting the similarity of the points observed in a pair of views. These methods are extremely powerful, because they are able to keep the statistical specificities of each view and to extract the complementary information from the diversity of the sources. Of particular interest is Multi-view Metric Learning (MVML [88]) which combines vvRKHS with Metric Learning [16, 212] and has proved to outperform Kernel-based state of the art methods, such as Multiple Kernel Learning [75].

MVML jointly learns a classifier and a kernel matrix encoding the within-view and between-view relationships. Although the computations are sped up by working on an approximated Gram matrix, obtained through the Nyström technique [206], this powerful approach is not sufficiently competitive in terms of execution time. However, this interesting idea faces a major issue: the cost required to extract the complementary information usually results in algorithms nonetheless barely competitive in terms of execution time.

To overcome the complexity burden of kernel-based methods, the approach **L^3-SVMs** presented in Chapter 5 for single-view classification is designed to take advantage of the discriminatory capabilities of kernels while being fast and scalable. Through clustering and projections on landmarks, this algorithm speeds up the learning process while training expressive classifiers, competitive with Kernel-SVMs. In this chapter, we aim at (i) benefiting from this promising landmarks-based SVM paradigm, (ii) adapting it to the multi-view scenario and (iii) deriving theoretical guarantees which take into account both the number of landmarks and views.

6.1.2 Missing View Imputation

Another open problem in multi-view learning is how to deal with realizations of the points that are partially incomplete, i.e. some views of certain instances are missing. In order to apply a multi-view algorithm, one might have to discard the points with missing views, which may result in a loss of performance, or to complete them using different techniques while trying not to introduce bias. Common practices consist in replacing the missing values with zeros or with the mean or the median values of the considered feature.

On the other hand, multi-view kernel specific techniques have been proposed to complete the Gram matrices of incomplete views. By making the assumption that similarities between points should be consistent from one view to another, the missing values of a view's Gram matrix are inferred by aligning its eigen-space to the ones of the other views. This can be done by Graph Laplacian regularization [186] (finding the matrix that minimizes its product with the Graph Laplacian matrix of a complete reference view) or by learning convex combinations of normalized kernel matrices [21]. The first limitation of such approaches comes from the fact that they cannot be applied on non-square matrices. This prevents us from using them on matrices containing the similarities to a subset of points, like in landmarks-based SVM approaches. Beyond this constraint, the assumption that views are strongly similar and the constraint of having the points altogether observed in a view seems too strong.

Another multi-view imputation technique relies on the existence of view generating functions for approximating the missing values. For example, in [2], the authors resort to translation functions for documents in multiple-languages. Unfortunately, depending on the application at hand, such functions are not always available.

In Section 6.4, we make use of the information coming from a small set of randomly selected landmarks to impute the missing values. As for Laplacian imputation [186], we do not need to reconstruct the actual missing features of a point, but only its similarities w.r.t. the landmarks, which drastically simplifies the problem. Through Least Square minimization, we impute the missing similarities by learning the linear combinations of the landmarks projected in the latent space.

6.2 Multi-View Landmark-based SVM (MVL-SVM)

Following the promising line of work of [88], in this Section we propose a new latent space-based approach, called **MVL-SVM**, which leverages the complementary information of the views and which is fast, scalable and provably effective. As shown in Fig.6.2, we base our work on Support Vector Machines (SVMs) [49]. In order to keep the time complexity and memory usage low, we formulate our problem as a Linear SVM in a joint space created by comparing the instances, a view at a time, to a small set of randomly selected landmarks, also observed in multiple views. The instance/landmark comparison is carried out by means of similarity functions, such as the RBF kernel, each defined on a view. Doing so, we solve a linearized joint problem over all views, in which the statistical characteristics of the views are recoded in similarity estimates with points spread over their spaces. Additionally, by applying non-linear mappings, we efficiently capture the non-linearities and multi-modalities of the view spaces while avoiding the drawbacks of kernel SVMs (see [7, 27, 107, 174]). Such benefits would not be possible without projecting the points on landmarks: the mapping ensures that the algorithm works on homogeneous features and it also controls the dimensionality of the projected space.

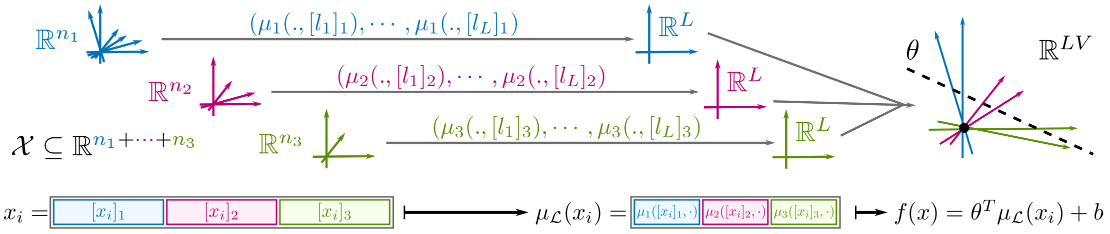

$\mathbb{R}^{n_1}$
$(\mu_1(., [l_1]_1), \cdots, \mu_1(., [l_L]_1)$
$\mathbb{R}^L$
$\mathbb{R}^{n_2}$
$(\mu_2(., [l_1]_2), \cdots, \mu_2(., [l_L]_2)$
$\mathbb{R}^L$
θ
$\mathbb{R}^{LV}$
$\mathcal{X} \subseteq \mathbb{R}^{n_1+\cdots+n_3}$
$\mathbb{R}^{n_3}$
$(\mu_3(., [l_1]_3), \cdots, \mu_3(., [l_L]_3)$
$\mathbb{R}^L$
$x_i =$ $[x_i]_1$ $[x_i]_2$ $[x_i]_3$
$\mu_{\mathcal{L}}(x_i) =$ $\mu_1([x_i]_1, \cdot)$ $\mu_2([x_i]_2, \cdot)$ $\mu_3([x_i]_3, \cdot)$
$f(x) = \theta^T \mu_{\mathcal{L}}(x_i) + b$

Figure 6.2 – *Overview of the proposed **MVL-SVM** method. From V views (3 here) of possibly different nature, points are projected on randomly selected landmarks $l_1, \cdots, l_L$ using view specific non-linear mappings $\mu_1, \cdots, \mu_V$. Then, a linear separator is learned in $\mathbb{R}^{LV}$, the joint space of projections.*

6.2.1 Notations and Problem Statement

We consider the problem of learning from a dataset $\mathcal{S} = \{z_i = (x_i, y_i)\}_{i=1}^m$ of m instances i.i.d. according to a joint distribution $\mathcal{D}$ and observed in a multi-view space of V views, so that $x_i \in \mathcal{X} \subseteq \mathbb{R}^{d_1+\cdots+d_V}$, in which views are potentially of different dimensionality and nature, and $y_i \in \mathcal{Y} = \{-1, 1\}$. In the following, we will use the notation $[x_i]_v$ to refer to the realization of point x_i in the view v. Moreover, we denote $\mathcal{L} = \{l_p\}_{p=1}^L \in \mathcal{X}^L$, a set of L landmarks of the input space selected randomly from the training sample.

We aim at learning a classifier $f : \mathcal{X} \to \mathbb{R}$ in the joint space defined by the different views as follows:

$$f(x) = \theta^T \mu_{\mathcal{L}}(x) + b \tag{6.1}$$

where $\theta \in \mathbb{R}^{LV}$ is a vector of weights, each associated to a view v of a landmark p and $\mu_{\mathcal{L}}(x_i) = [\mu_1([x_i]_1, [l_1]_1), \ldots, \mu_1([x_i]_1, [l_L]_1), \ldots, \mu_V([x_i]_V, [l_L]_V)]$ can be interpreted as the mapping function from the input space $\mathcal{X}$ to a new landmark space $\mathcal{H} \subseteq \mathbb{R}^{LV}$. The sign of the function is used for prediction ($\hat{y} = sign(f(x))$), i.e. test examples need to be projected as well on the latent space. Notice that each point is compared to the set of landmarks one view at a time and that the problem is now linear in the space $\mathcal{H}$. To capture the non-linearities of the space, we rely on the choice of view-specific score functions $\mu_v : \mathbb{R}^{n_v} \times \mathbb{R}^{n_v} \to \mathbb{R}$ between representations of points in a given view.

The choice of projecting the dataset on selected landmarks is crucial for the discriminatory power of the resulting classifier. As a matter of fact, it enables to express the statistical peculiarities of a view-space through similarity estimates and additionally it allows us to work on a latent space common to all views, which has multiple benefits: firstly, it allows to control the dimensionality of the space by choosing the number of landmarks; secondly, it enables learning of a unique classifier, avoiding the problem of combining the outputs of view-specific models; lastly, and most importantly, it loosens the assumptions on the relationship between view information, especially the one about their correlation.

6.2.2 Optimization Problem and Algorithm

As for standard SVMs, our objective function consists in maximizing the margin between the class hyperplanes while minimizing a surrogate function of the classification error:

Algorithm 2: MVL-SVM algorithm.

```
Input: a sample $\mathcal{S} = \{z_i = (x_i, y_i)\}_{i=1}^m \subseteq \mathbb{R}^{n_1+\cdots+n_V} \times \{-1, 1\}$
and a set of view-specific score functions $\{\mu_v : \mathbb{R}^{n_v} \times \mathbb{R}^{n_v} \to \mathbb{R}\}_{v=1}^V$
1. Select $\mathcal{L} = \{l_p\}_{p=1}^L$ uniformly from $\{x_i\}_{i=1}^m$;
2. Project $\mathcal{S}$ on the latent space:
for $i = 1$ to $m$ do
    $\mu_{\mathcal{L}}(x_i) = [\mu_1([x_i]_1, [l_1]_1), \ldots, \mu_1([x_i]_1, [l_L]_1), \ldots, \mu_V([x_i]_V, [l_L]_V)]$
end for
3. Learn $\theta \in \mathbb{R}^{LV}$ as the minimizer of Problem (6.2.2);
4. Use $sign(\theta^T \mu_{\mathcal{L}}(x) + b)$ for prediction.
```

$$F(f) = \frac{1}{2} \|f\|^2 + \frac{c}{m} \sum_{i=1}^{m} \ell(f, z_i) \tag{6.2}$$

where $\ell(f, z) = \max(0, 1 - yf(x))$ is the hinge loss. We formulate the multi-view classification problem as a soft-margin SVM learning that we solve in its primal form:

$$\begin{aligned} &\arg\min_{\theta,b,\xi} \frac{1}{2} \|\theta\|^2 + \frac{c}{m} \sum_{i=1}^{m} \xi_i \\ s.t.\ \ & y_i \left(\theta^T \mu_{\mathcal{L}}(x_i) + b\right) \geq 1 - \xi_i \ ;\ \xi_i \geq 0 \ \forall i = 1..m. \end{aligned}$$

The main difference with a standard SVM is the working input space and its interpretation. Basically, we learn how to linearly combine the point-landmark similarities, describing how they should change over the views for a class. The pseudo-code of **MVL-SVM** is reported in Algorithm 2.

To recapitulate, our landmark-induced latent space allows us to efficiently extract the complementarity between views while capturing their statistical peculiarities. Moreover, **MVL-SVM**'s flexibility makes it suited to deal with multiple and not-necessarily correlated views, potentially heterogeneous and of different dimensionality. This flexibility, combined with its scalability, makes **MVL-SVM** applicable to a wide set of problems.

6.3 Theoretical results

Since the parameters θ and b are optimized from a finite set of training examples, a key question is how the learned model behaves at test time. Using the theoretical framework of the Uniform Stability [30] presented in Chapter 1, we analyze in this section the generalization properties of our algorithm by deriving an upper bound on its true risk. We will see that the stability of our method and, consequently, its generalization capabilities, depend on the choice of the projection functions, the number of selected landmarks and the characteristic of the dataset, such as the number of views and the size of the training set.

Here, we skip most of the notions common with the derivation of Chapter 5. We invite the reader to refer to Chapter 1 and 5 for more details.

6.3.1 MVL-SVM's Uniform Stability

Let S be the original dataset and $S^{\setminus i}$ the set obtained after removing the i^{th} sample z_i from S. We denote by f (resp. $f^{\setminus i}$) the optimal solution of (6.2.2) on S (resp. $S^{\setminus i}$). Moreover, we recall that, as noted in [30], the hinge loss is 1-admissible.

Theorem 6.1 **Uniform Stability** *Given the inverse regularizer weight c (from Eq. (6.2.2)), **MVL-SVM** has uniform stability $\frac{cLVM^2}{m}$, where $M = 1$ if μ_v uses the RBF kernel $\forall v$.*

Proof. As $\ell(f, z)$ is 1-admissible, $\forall z = (x, y) \in \mathcal{Z}$,

$$\left|\ell(f^{\setminus i}, z) - \ell(f, z)\right| \leq \left|f^{\setminus i}(x) - f(x)\right| = |\Delta f(x)| \tag{6.3}$$

with $\Delta f = f^{\setminus i} - f$. By denoting $\Delta\theta = \theta^{\setminus i} - \theta$, we can derive, $\forall z = (x, y) \in \mathcal{Z}$,

$$\begin{aligned}
|\Delta f(x)| &= \left|\theta^{\setminus i}\mu_{\mathcal{L}}(x)^T - \theta\mu_{\mathcal{L}}(x)^T\right| \\
&= \left|(\theta^{\setminus i} - \theta)\mu_{\mathcal{L}}(x)^T\right| \\
&\leq \left\|\theta^{\setminus i} - \theta\right\| \|\mu_{\mathcal{L}}(x)\| \qquad (6.4) \\
&\leq \|\Delta\theta\| \|\mu_{\mathcal{L}}(x)\| \\
&\leq \|\Delta\theta\| \sqrt{LV} \|\mu_{\mathcal{L}}(x)\|_\infty \qquad (6.5) \\
&\leq \|\Delta\theta\| \sqrt{LV} \max_{l,v}(\mu_v([x]_v, [l]_v)) \\
&\leq \|\Delta\theta\| \sqrt{LV} M \qquad (6.6)
\end{aligned}$$

with $M = \max_{l,v}(\mu_v([x]_v, [l]_v))$.

Eq. (6.4) is due to the Cauchy-Swartz inequality and Eq. (6.5) is because $\|\mu_{\mathcal{L}}(x)\| \leq \sqrt{LV} \|\mu_{\mathcal{L}}(x)\|_\infty$ recalling that $\mu_{\mathcal{L}}(x) \in \mathbb{R}^{LV}$.

The value of M depends on the chosen score functions $\{\mu_v\}_{v=1}^V$. For instance, if every μ_v is an RBF kernel, then $M = 1$.

From Lemma 21 of [30] we get:

$$2\left\|\Delta\theta\right\|^2 \leq \frac{c}{m}\left|\Delta f(x_i)\right|.$$

Then, by instantiating Eq. (6.6) for $x = x_i$, we get

$$\left\|\Delta\theta\right\|^2 \leq \frac{c}{2m}\left|\Delta f(x_i)\right| \leq \frac{c}{2m}\left\|\Delta\theta\right\|\sqrt{LV}M$$

and as $\|\Delta\theta\| > 0$, we obtain

$$\|\Delta\theta\| \leq \frac{c}{2m}\sqrt{LV}M. \tag{6.7}$$

Finally, plugging Eq. (6.7) in Eq. (6.6), we get

$$\forall z = (x, y),\quad \left|\Delta f(x)\right| \leq \left\|\Delta\theta\right\|\sqrt{LV}M \leq \frac{cLVM^2}{2m}$$

which, with Eq. (6.3), gives the $\frac{cLVM^2}{m}$ uniform stability.

□

Note that the stability of **MVL-SVM** depends on the number of landmarks L. Our method is stable only if $L \ll \frac{m}{V}$, which is not a strong condition considering that usually $m \gg V$. Moreover, this bound expresses that, the smaller the L, the more stable the algorithm. This is consistent with the fact that L controls the dimensionality of the projected space in which the multi-view model is learned.

6.3.2 Generalization Bound

The following results applies for upper bounded losses, such that $0 \leq \ell(f, z) \leq E$ with $E \in \mathbb{R}^+$.

Corollary 6.1 *The generalization bound of **MVL-SVM** derived using the Uniform Stability framework is as follows:*

$$R_{\mathcal{D}}(f) \leq \hat{R}_S(f) + \frac{cLVM^2}{m} + \Big(2cLVM^2 + 1 + 2c\sqrt{LV}M\Big)\sqrt{\frac{\ln\frac{1}{\delta}}{2m}}.$$

Proof. The constant E can be estimated by considering the following:

$$
\begin{aligned}
F(f) &\leq F(\mathbf{0}) \\
\frac{1}{2}\left\|\theta\right\|^2 + \frac{c}{m}\sum_{i=1}^{m}\max(0, 1-y_i(\theta\mu_{\mathcal{L}}(x_i)^T)) &\leq \frac{1}{2}\left\|\mathbf{0}\right\|^2 + \frac{c}{m}\sum_{i=1}^{m}\max(0, 1-y_i(\mathbf{0}\mu_{\mathcal{L}}(x_i)^T)) \\
\frac{1}{2}\left\|\theta\right\|^2 &\leq c \qquad (6.8) \\
\left\|\theta\right\|^2 &\leq 2c
\end{aligned}
$$

Eq. (6.8) is because $\forall a, b, c \in \mathbb{R}^+$, $a + b \leq c$ implies that $b \leq c$. Thus,

$$
\begin{aligned}
\ell(f, z) &= \max(0, 1 - y\theta\mu_{\mathcal{L}}(x)^T) \\
&\leq 1 + \left|\theta\mu_{\mathcal{L}}(x)^T\right| \\
&\leq 1 + \left\|\theta\right\| \left\|\mu_{\mathcal{L}}(x)\right\| \qquad (6.9) \\
&\leq 1 + 2c\sqrt{LV}M = E
\end{aligned}
$$

Eq. (6.9) comes again from the Cauchy-Swartz inequality. □

The main difference with the bound of the previous chapter lies in the dependence to the number of views V.

6.4 Learning with missing views

Up to this section, we have made the implicit assumption that all the instances were observed in all the views. Because it is common in real-case scenarios that some points are observed only in a subset of views, we now illustrate how to adapt our formulation to this so-called missing-view setting.

The formulation from Eq. 6.2.2 is applicable only when all the points of the training and test sets are observed in all the views. To extend our method to the context of missing views, we apply a reconstruction step before learning. As we want to preserve the scalability of our approach, we do not impute missing values in the original input space: we rather design a dedicated method that imputes missing values by directly leveraging the information coming from the set of landmarks. We simply formulate our imputation as a Least Square estimation over the known values as follows:

$$
\arg\min_{R} \left\|M - RP\right\|_{\mathcal{F}'}^2 \qquad (6.10)
$$

with M the $m \times LV$ matrix of projection values, P the $L \times LV$ matrix of projected landmarks, R the unknown $m \times L$ reconstruction matrix and $\|.\|_{\mathcal{F}'}$ the Frobenius norm considering only the non-missing values (in our case, the missing values are those of M). The problem from Eq. (6.10) boils down to learning linear combinations of landmark similarities over all the views and, for this reason, all the views of the landmarks need to be known. Doing so, we avoid estimating the actual missing features and we directly impute the view-dependent similarities between points and landmarks.

It is worth noting that each point projection is reconstructed independently and that the system is always (over-)determined for each point, as at least one block of size L of the point projection is known (at least one view's features are given) and the number of unknowns is L.

6.5 Experimental results

In this section, we report and analyze the performances of our method w.r.t. the state of the art algorithms, in terms of both classification accuracy and training and testing execution times. We perform two sets of experiments: (i) learning with complete views and (ii) learning with missing views. We will specifically study the behavior of **MVL-SVM** w.r.t. the number of landmarks keeping in mind that the larger the number of landmarks, the better the discriminatory power of the classifier, but the slower the learning process.

An implementation of our method, based on the Liblinear library [60], together with the other existing algorithms (when the codes are open-source) is available at https://github.com/vzantedeschi/multiviewLSVM.

6.5.1 Datasets, Methods and Experimental Setup

For these experiments, we employ two multi-class datasets that provide multi-view representations of the instances:

- Flower17 [2] contains 1360 pictures of 17 categories of flowers, which come with 7 different distance matrices between pictures (*i.e.* the 7 views);
- uWaveGesture [44] is formed by 4478 vectors describing 8 different gestures as captured by 3 accelerometers (the 3 views).

In order to prove the significance of embedding the datasets in a single space, we compare our technique to the methods that learn a single classifier on a latent

[2] http://www.robots.ox.ac.uk/$\sim$vgg/data/flowers/17/

space and to the methods that learn a set of single-views classifiers. Moreover, we principally compare **MVL-SVM** to SVM-based approaches, to highlight the interest of using landmark-mappings. Multi-class classification is carried-out through the one-vs-all procedure.

We report the results of the following baselines:

- **MVML** [88] that optimizes over both the classifier and the metric matrix, and which is designed to make the most of the between-view and within-view relationships;
- the co-regularization technique **SVM-2k** [61], which regularizes over the predictions enforcing their smoothness. Originally designed for 2 view learning, we adapted this algorithm to work with $V \geq 2$ views by learning a SVM-2k for every pair of views and combining their predictions using a majority vote;
- **SVMs** which consists in learning a Kernel-SVM per view and aggregating their predictions by majority vote.

All the previous methods, and ours, utilize the Radial Basis Function (RBF, squared exponential) kernel for comparing the points, with a radius that is fixed and equals to the square root of the number of features. We make use of the 3 train-validation-test splits provided for Flower17, and of the train-test split for uWaveGesture, by tuning with cross-validation over the training set. We repeat each experiment 5 times, reporting the average test value and its standard deviation when it is not null. For **MVL-SVM**, at each iteration we randomly select a new set of landmarks to underline how the chosen landmarks affect the expressiveness of the latent space. We tune the hyper-parameters of the methods by grid-search over the following set-values: for MVML, we evaluate $\lambda \in \{10^{-8}, \dots, 10\}$ and $\eta \in \{10^{-3}, \dots, 10^{2}\}$, as indicated in the original paper; for SVM-2K, we consider c_1, c_2 and $d \in \{10^{-4}, \dots, 1\}$ and fix $\varepsilon = 10^{-3}$; for both SVMs and **MVL-SVM**, we consider $c \in \{10^{-3}, \dots, 10^{4}\}$.

6.5.2 Learning with Complete Views

In this first experiment, we compare the methods on complete datasets, where all the points are observed on all the views. In particular, we study the impact of the dimensionality of our latent space, controlled by the number of landmarks, on the performances of **MVL-SVM**. As the rank of the Nyström-approximated Gram matrix of MVML and the number of landmarks of **MVL-SVM** are comparable, because they both measure the number of computed similarities, we draw them on the same axis and compare these two methods also on this criterion. We explore values ranging from 10 to the size of the training set (validation set not included).

Because of MVML's huge computational complexity (see Fig. 6.4), its results in Figure 6.3 are truncated at a smaller approximation level.

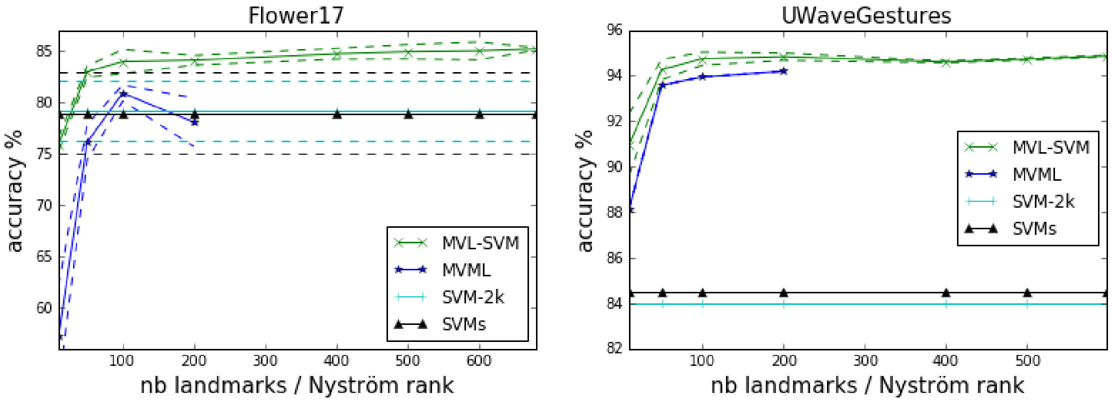


Figure 6.3 – *Average test accuracies (with standard deviations) w.r.t. the number of landmarks/Nyström rank.*

Figure 6.3 shows the test accuracies on both datasets. It is manifest how working on a latent space is of great benefit: both methods that exploit this idea show significant better test accuracies than those that learn view-specific classifiers, especially for the uWaveGestures dataset where views are very complementary. It is worth noting that **MVL-SVM** is able to reach the best performance even with a small number of landmarks (10 for uWaveGestures and 50 for Flower17).

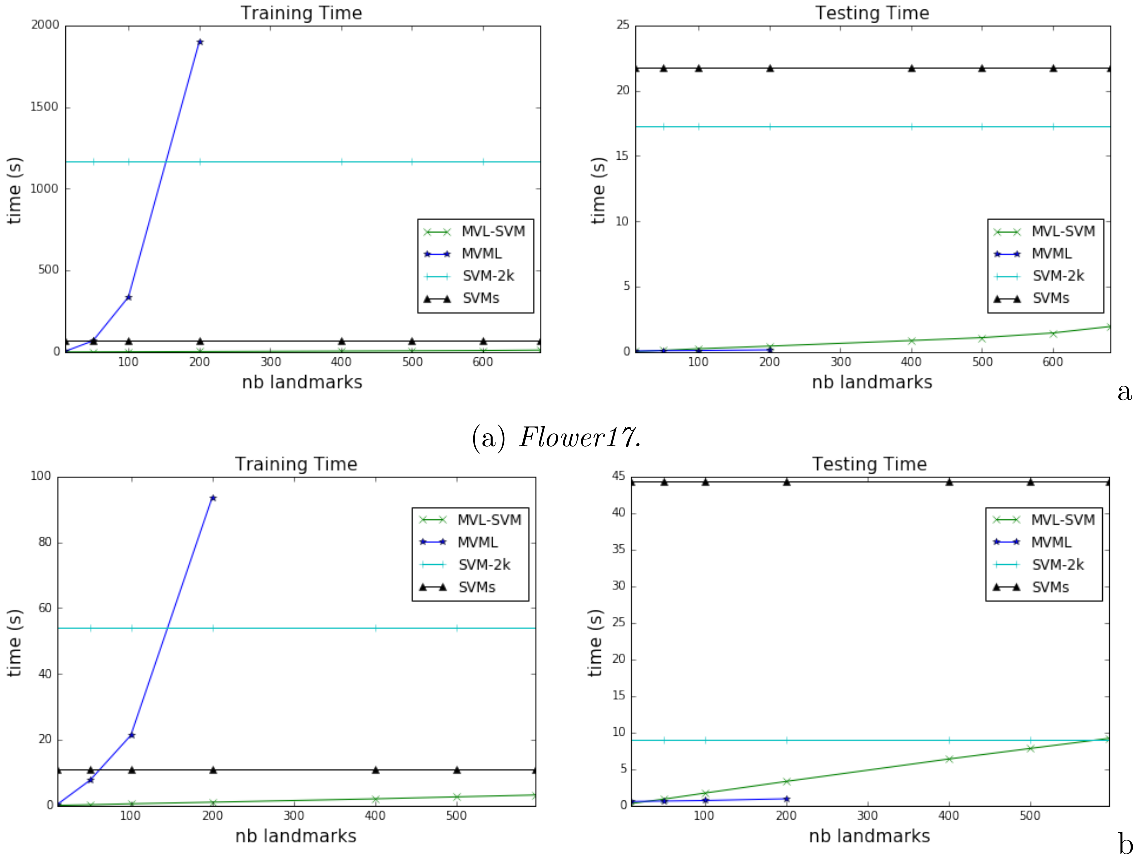
Moreover, majority-vote techniques seem more sensitive to the choice of points selected for training (see Flower17) than **MVL-SVM**, which is consistently robust to the variations in the set of landmarks.

Figures 6.4 and 6.5 highlight the other important advantage of **MVL-SVM**: its fastness. At training time, **MVL-SVM**'s execution time is linear in the number of landmarks and several magnitudes smaller than those obtained for the baselines. At test time, **MVL-SVM** is only slightly beaten by MVML, but it is probably due to better optimizations in the code. Notice how learning multiple learners (SVM-2k and SVMs) considerably slows down both the training and the test steps. Handling multiple models is, indeed, a heavy overhead.

Overall, **MVL-SVM** achieves significantly better test accuracy that the considered baselines, even with a limited number of landmarks, while training several order of magnitude faster and being comparably fast at test time.

6.5.3 Learning with Missing Views

With this second series of experiments, we aim to evaluate the validity of the imputation technique proposed in Section 6.4. We make use of the two previously described datasets that we modify for the current task: we drop random views of



(a) *Flower17.*

(b) *uWaveGestures.*

Figure 6.4 – *Training and testing times w.r.t. the number of landmarks.* ***MVL-SVM*** *is very fast and scales linearly with the number of landmarks, unlike MVML.*

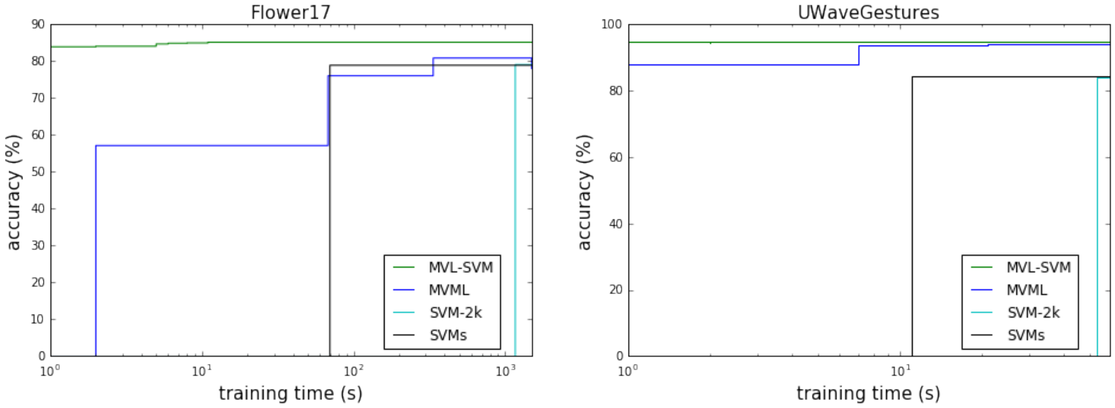


Figure 6.5 – *Average test accuracies w.r.t. the training time. Compared to the other methods,* ***MVL-SVM*** *reaches high accuracy even with very low computational budget. The x axis is in logarithmic scale.*

their points with a ratio of missing views over total number of views (mV) which we vary over the interval $[0, 0.5]$. For **MVL-SVM**, the number of landmarks L is fixed to 200. In Figure 6.6, we plot the test accuracies in this new setting for both datasets, comparing **MVL-SVM** to **SVMs** with and without any reconstruction technique. When no imputation is applied as preprocessing, the points with missing views are dropped for **MVL-SVM**, while for **SVMs**, as it deals with a view at a time, they are still used for training the view-specific models corresponding to the available views. For **SVMs**, we impute the missing values using Graph Laplacian imputation [186] by fixing the Gram matrix of the view with the most points as the reference view for reconstructing all the other views. Remark that the points missing from the reference view will not have their views reconstructed, which might explain the drop in accuracy of **SVMs** for a ratio bigger than 0.3 for Flower17.

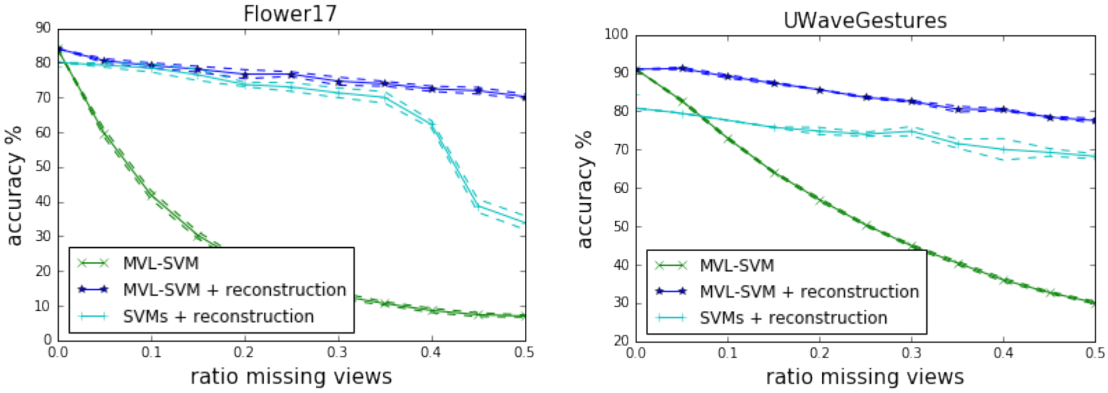


Figure 6.6 – *Test accuracies (with standard deviations) w.r.t. the ratio of missing views, using 200 landmarks for* ***MVL-SVM****. Imputation of missing value is critical for* ***MVL-SVM*** *to achieve good accuracy when facing missing views. Thanks to the proposed missing value imputation,* ***MVL-SVM*** *remains more accurate even in the case of missing views.*

Notice how preprocessing the dataset is fundamental for applying **MVL-SVM** to the missing-view scenario. This is not surprising as, using a latent space, we can train the model only on points observed in all the views. Even if the accuracy of both methods (with reconstruction) decays slightly with the ratio of missing views, the gain in performances is dramatic.

6.6 Conclusion

We proposed **MVL-SVM**, an effective technique for tackling multi-view problems, by training a linear-SVM on a landmark-induced latent space, which unifies the view information, constructed by applying non-linear multi-view similarity estimates between the instances and a set of randomly selected landmarks. We additionally introduced an imputation technique making it suitable for the missing-view context. We also showed **MVL-SVM**'s validity, from both theoretical and empirical point of view: we derived a generalization bound using the uniform stability framework, and we showed empirically that our approach outperforms the considered baselines in terms of accuracy while being several order of magnitude faster.

Apart from the generic perspectives of **L^3-SVMs** cited in Chapter 5, we identified several lines of work specific to the multi-view scenario. **MVL-SVM** relies on a set of landmarks that is shared across all views. According to the application at hand, it might be interesting to consider more landmarks in some of the views, and future work will include this consideration of different landmarks in the views. Additionally, by using block-sparsity in the final linear-separator, automated landmark selection could be achieved, giving **MVL-SVM** an even better test-time execution speed. The missing view imputation technique can also be improved by considering a joint optimization of the reconstruction matrix R and the linear classifier (θ, b).

Conclusion

In this manuscript, we tackled the problem of learning models capable of capturing local characteristics of data distribution, with a focus on their generalization, smoothness in prediction and scalability. We contributed in two parallel lines of research in local learning: we proposed two approaches based on data partitioning and an algorithm based on landmark similarities, adapted to single-view and multi-view data. Thorough studies were conducted to highlight the effectiveness of the said contributions which confirmed the soundness of their intuitions. We empirically studied the performance of the proposed methods both on toy and real tasks, in terms of accuracy and execution time, and compared it to state of the art results. We also analyzed our approaches from a theoretical standpoint, by studying their computational and memory complexities and by deriving tight generalization bounds.

Apart from the perspectives specific to each contribution that we evoked in the respective chapters, we consider several additional avenues for further research. Regarding the regularization term used in Chapters 3 and 4 on the pair-wise similarities between learned local models, we ponder new ways of estimating the similarity graph, i.e. the penalization weights affected to each pair of models. It would be interesting to optimize the graph as done in Chapter 4, but to drop the uniformity constraint on the degrees and number of neighbors of each node. A less constrained formulation would have the advantage of better capturing the relationships between local models and of detecting communities of users when working with personal data.

Another attractive perspective would be to study the works of Part III under the $(\epsilon, \gamma, \tau)-$good similarities framework presented in Chapter 2. Indeed, this framework offers a principled way for studying the quality of the linearization of the problem induced by a mapping based on landmark similarities, like in our works. Yet, it would not be straightforward to theoretically analyze our contributions through this theory, as it stands for landmarks drawn i.i.d. from the data distribution and for bounded similarity functions. These assumptions are not necessarily satisfied in our works: we admit the use of virtual landmarks and of unbounded kernels such as the linear kernel. Moreover, we believe that restraining the similarity function to be a Mercer's kernel should lead to better theoretical results.

Furthermore, we are particularly interested in methods for drawing good sets of landmarks, in the sense that they would be minimal but representative of the data distribution, and to study the trade-off between accuracy and scalability, depending on the complexity of the task. We argue that such a study on the effects of the quality of the sample can lead to tighter generalization bounds.

Finally, it would be insightful to assess the robustness to adversarial examples of our methods, using the measures reported in Chapter 1. As a matter of fact, we are intrigued about the higher robustness observed in other models that learn latent spaces w.r.t. models that do not.

Part IV

Appendices

A β-risk: a New Surrogate Risk for Learning from Weakly Labeled Data

This Appendix reports the publication

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. Beta-risk: a new surrogate risk for learning from weakly labeled data. In *Advances in Neural Information Processing Systems*, pages 4365–4373, 2016.

During the past few years, the machine learning community has paid attention to developing new methods for learning from weakly labeled data. This field covers different settings like semi-supervised learning, learning with label proportions, multi-instance learning, noise-tolerant learning, etc. This paper presents a generic framework to deal with these weakly labeled scenarios. We introduce the β-risk as a generalized formulation of the standard empirical risk based on surrogate margin-based loss functions. This risk allows us to express the reliability on the labels and to derive different kinds of learning algorithms. We specifically focus on SVMs and propose a soft margin β-SVM algorithm which behaves better that the state of the art.

A.1 Introduction

The growing amount of data available nowadays allowed us to increase the confidence in the models induced by machine learning methods. On the other hand, it also caused several issues, especially in supervised classification, regarding the availability of labels and their reliability. Because it may be expensive and tricky to assign a reliable and unique label to each training instance, the data at our disposal for the application at hand are often *weakly labeled*. Learning from weak supervision has received important attention over the past few years [91, 118]. This research field includes different settings: only a fraction of the labels are known (Semi-Supervised learning [229]); we can access only the proportions of the classes (Learning with Label Proportions [154] and Multi-Instance Learning [54]); the labels are uncertain or noisy (Noise-Tolerant Learning [3, 141, 153]); different discording labels are given to the same instance by different experts (Multi-Expert Learning [169]); labels are completely unknown (Unsupervised Learning [83]). As a consequence of this statement of fact, the data provided in all these situations cannot be fully exploited using supervised techniques, at the risk of drastically reducing the performance of the learned models. To address this issue, numerous machine learning methods have been developed to deal with each of the previous specific situations. However, all these weakly labeled learning tasks share common features mainly relying on the confidence in the labels, opening the door to the development of generic frameworks. Unfortunately, only a few attempts have tried to address several settings with the same approach. The most interesting one has been presented in [118] where the authors propose WellSVM which is dedicated to deal with three different weakly labeled learning scenarios: semi-supervised learning, multi-instance learning and clustering. However, WellSVM focuses specifically on Support Vector Machines and it requires to derive a new optimization problem for each new task. Even though WellSVM constitutes a step further towards general models, it stopped in midstream constraining the learner to use SVMs.

This paper aims to bridge this gap by presenting a generic framework for learning from weakly labeled data. Our approach is based on the derivation of the β-risk , a new surrogate empirical risk defined as a strict generalization of the standard empirical risk relying on surrogate margin-based loss functions. The main interesting property of the β-risk comes from its ability to exploit the information given by the weakly supervised setting and encoded as a β matrix reflecting the supervision on the labels. Moreover, the instance-specific weights β let one integrate in classical methods the side information provided by the setting. This is the peculiarity w.r.t. [141, 153]: in both papers, the proposed losses are defined using class-dependent weights (fixed to 1/2 for the first paper, and dependent on the class noise rate for the latter) while in our approach the used weights are provided for each instance, which gives a more flexible formulation. Making use of this β-risk , we design a generic algorithm devoted to address different kinds of aforementioned weakly labeled settings. To allow a comparison with the state of the art, we instantiate it with a learner that takes the form of an SVM algorithm. In this context, we derive a soft margin β-SVM algorithm and show that it outperforms WellSVM.

The remainder of this paper is organized as follows: in Section A.2, we define the empirical surrogate β-risk and show under which conditions it can be used to learn without explicitly accessing the labels; we also show how to instantiate β according to the weakly labeled learning setting at hand; in Section A.3, we present our generic iterative algorithm for learning with weakly labeled data and in Section A.8.3 we exploit our new framework to derive a novel formulation of the Support Vector Machine problem, the β-SVM ; finally, we report experiments in semi-supervised learning and learning with label noise, conducted on classical datasets from the UCI repository [120], in order to compare our algorithm with the state of the art approaches.

A.2 From classical surrogate losses and surrogate risks to the β-risk

In this section, we first provide reminders about surrogate losses and then exploit the characteristics of the popular loss functions to introduce the empirical surrogate β-risk . The β-risk formulation allows us to tackle the problem of learning with weakly labeled data. We show under which conditions it can be used instead of the standard empirical surrogate risk (defined in a fully supervised context). Those conditions give insight on how to design algorithms that learn from weak supervision. We restrain our study to the context of binary classification.

A.2.1 Preliminaries

In statistical learning, a common approach for choosing the optimal hypothesis h^* from a hypothesis class $\mathcal{C}$ is to select the classifier that minimizes the expected risk over the joint space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the feature space and $\mathcal{Y}$ the label space, expressed as

$$\mathcal{R}_\ell(h) = \int_{X \times Y} \ell(yh(x))p(x,y)dxdy$$

with $\ell : \mathcal{C} \times Z \to \mathbb{R}^+$ a margin-based loss function.

Since the true distribution of the data $p(x,y)$ is usually unknown, machine learning algorithms typically minimize the empirical version of the risk, computed over a finite set S composed of m instances (x_i, y_i) i.i.d. drawn from a distribution over $\mathcal{X} \times \{-1, 1\}$:

$$\mathcal{R}_\ell(S,h) = \frac{1}{m}\sum_{i=1}^{m} \ell(y_i h(x_i)).$$

The most natural loss function is the so-called 0-1 loss. As this function is not convex, not differentiable and has zero gradient, other loss functions are commonly employed instead. These losses, such as the logistic loss (e.g., for the logistic regression [46]), the exponential loss (e.g., for boosting techniques [69]) and the hinge loss (e.g., for the SVM [49]), are convex and smooth relaxations of the 0-1 loss. Theoretical studies on the characteristics and behavior of such *surrogate losses* can be found in [19, 146, 159]. In particular, [146] showed that each commonly used surrogate loss can be characterized by a permissible function ϕ (see below) and rewritten as $F_\phi(x)$

$$F_\phi(x) = \frac{\phi^*(-x) - a_\phi}{b_\phi}$$

where $\phi^*(x) = \sup_a(xa - \phi(a))$ is the Legendre conjugate of ϕ (for more details, see [32]), $a_\phi = -\phi(0) = -\phi(1) \geq 0$ and $b_\phi = -\phi(\frac{1}{2}) - a_\phi > 0$. As presented by the authors of [95] and [146], a permissible function is a function $f : [0,1] \to \mathbb{R}^-$, symmetric about $-\frac{1}{2}$, differentiable on $]0,1[$ and strictly convex. For instance, the permissible function ϕ_{log} related to the logistic loss $F_\phi(x) = \log(1 + \exp^{-x})$ is:

$$\phi_{log}(x) = x\log(x) + (1-x)\log(1-x)$$

and $a_\phi = 0$ and $b_\phi = \log(2)$.

As detailed in [146], considering a surrogate loss F_ϕ, the empirical surrogate risk of an hypothesis $h : X \to \mathbb{R}$ w.r.t. S can be expressed as:

$$\mathcal{R}_\phi(S,h) = \frac{1}{m}\sum_{i=1}^{m} D_\phi\Big(y_i, \nabla_\phi^{-1}(h(x_i))\Big) = \frac{b_\phi}{m}\sum_{i=1}^{m} F_\phi(y_i h(x_i))$$

with D_ϕ the Bregman Divergence

$$D_\phi(x,y) = \phi(x) - \phi(y) - (x-y)\nabla_\phi(y).$$

In order to evaluate such risk $\mathcal{R}_\phi(S,h)$, it is mandatory to provide the labels y for all the instances. In addition, it is not possible to take into account eventual uncertainties on the given labels. Consequently, $\mathcal{R}_\phi$ is defined in a totally supervised context, where the labels y are known and considered to be true. In order to face the numerous situations where training data may be weakly labeled, we claim that there is a need to fill the gap by defining a new empirical surrogate risk that can deal with such settings. In the following section, we propose a generalization of the empirical surrogate risk, called the empirical surrogate $\beta-$risk, which can be employed in the context of weakly labeled data instead of the standard one under some linear conditions on the margin.

A.2.2 The Empirical Surrogate β-risk

Before defining the empirical surrogate β-risk for any loss F_ϕ and hypothesis $h \in \mathcal{C}$, let us rewrite the definition of $\mathcal{R}_\phi$ introducing a new set of variables named β, and that can be laid out as a $2\times m$ matrix.

Lemma A.1 *For any S, ϕ and h, and for any non-negative real coefficients β_i^{-1} and β_i^{+1} defined for each instance $x_i \in S$ such that $\beta_i^{-1} + \beta_i^{+1} = 1$, the empirical surrogate risk $\mathcal{R}_\phi(S,h)$ can be rewritten as*

$$\mathcal{R}_\phi(S,h) = \mathcal{R}_\phi(S,h,\beta)$$

where

$$\mathcal{R}_\phi(S,h,\beta) = \frac{b_\phi}{m} \sum_{i=1}^{m} \sum_{\substack{\sigma\in \\ \{-1,+1\}}} \beta_i^{\sigma} F_\phi(\sigma h(x_i)) + \frac{1}{m} \sum_{i=1}^{m} \beta_i^{-y_i}(-y_i h(x_i)).$$

The coefficient β_i^{+1} (resp. β_i^{-1}) for an instance x_i can be interpreted here as the degree of confidence in (or the probability of) the label +1 (resp. -1) assigned to x_i.

Proof.

$$\begin{aligned}
\mathcal{R}_\phi(S,h) &= \frac{b_\phi}{m} \sum_{i=1}^{m} F_\phi(y_i h(x_i)) \\
&= \frac{b_\phi}{m} \sum_{i=1}^{m} \Big(\beta_i^{y_i} F_\phi(y_i h(x_i)) + \beta_i^{-y_i} F_\phi(y_i h(x_i))\Big) && \text{(A.1)} \\
&= \frac{b_\phi}{m} \sum_{i=1}^{m} \left(\beta_i^{y_i} F_\phi(y_i h(x_i)) + \beta_i^{-y_i} \left(F_\phi(-y_i h(x_i)) - \frac{y_i h(x_i)}{b_\phi}\right)\right) && \text{(A.2)} \\
&= \frac{b_\phi}{m} \sum_{i=1}^{m} \sum_{\substack{\sigma\in \\ \{-1,+1\}}} \beta_i^{\sigma} F_\phi(\sigma h(x_i)) + \frac{1}{m} \sum_{i=1}^{m} \beta_i^{-y_i}(-y_i h(x_i)). && \text{(A.3)}
\end{aligned}$$

Eq. (A.1) is because $\beta_i^{-1} + \beta_i^{+1} = 1$; Eq. (A.2) is due to the fact that $\phi^*(-x) = \phi^*(x) - x$ (see the supplementary material) for any permissible function ϕ, so that $F_\phi(x) = \frac{\phi^*(-x)-a_\phi}{b_\phi} = \frac{\phi^*(x)-a_\phi-x}{b_\phi} = F_\phi(-x) - \frac{x}{b_\phi}$. □

From Eq. (A.3), and considering that the sample S is composed by the finite set of features $\mathcal{X}$ and labels $\mathcal{Y}$, we can write that

$$\mathcal{R}_\phi(S,h) = \mathcal{R}_\phi(S,h,\beta) = \mathcal{R}_\phi^{\beta}(\mathcal{X},h) - \frac{1}{m} \sum_{i=1}^{m} \beta_i^{-y_i} y_i h(x_i) \tag{A.4}$$

where

$$\mathcal{R}_\phi^{\beta}(\mathcal{X},h) = \frac{b_\phi}{m} \sum_{i=1}^{m} \sum_{\substack{\sigma\in \\ \{-1,+1\}}} \beta_i^{\sigma} F_\phi(\sigma h(x_i))$$

is the empirical surrogate β-risk for a matrix $\beta = [\beta_0^{+1}, ..., \beta_m^{+1} | \beta_0^{-1}, ..., \beta_m^{-1}]$.

It is worth noticing that $\mathcal{R}_\phi(S,h,\beta)$ is expressed in the form of a sum of two terms: the second one takes into account the labels of the data, while the first one, the β-risk, focuses on the loss suffered by h over $\mathcal{X}$ without explicitly needing the labels $\mathcal{Y}$.

The empirical β-risk is a generalization of the empirical risk: setting $\beta_i^{y_i} = 1$ (and thus $\beta_i^{-y_i} = 0$) for each instance, the second term vanishes and we retrieve the classical formulation of the empirical risk. Additionally, as developed in Section A.2.3, the introduction of β makes it possible to inject some side-information about the labels. For this reason, we claim that the β-risk is suited to deal with classification in the context of weakly labeled data.

Let us now focus on the conditions allowing the empirical β-risk (i) to be a surrogate of the 0-1 loss-based empirical risk and (ii) to be sufficient to learn with a weak supervision on the labels. From (A.4), we deduce:

$$\mathcal{R}_\phi^\beta(\mathcal{X}, h) = \mathcal{R}_\phi(S, h, \beta) + \frac{1}{m}\sum_{i=1}^{m} \beta_i^{\text{-}y_i} y_i h(x_i) \geq \mathcal{R}_{0/1}(S, h) + \frac{1}{m}\sum_{i=1}^{m} \beta_i^{\text{-}y_i} y_i h(x_i) \tag{A.5}$$

where $\mathcal{R}_{0/1}(S, h)$ the empirical risk related to the 0-1 loss and Eq. (A.5) is because $b_\phi F_\phi(x) \geq F_{0/1}(x)$ (for any surrogate loss).

It is possible to ensure that the β-risk is both a convex upper-bound of the 0-1 loss based risk and a relaxation as tight as the traditional risk (i.e., that we have $\mathcal{R}_{0/1}(S, h) \leq \mathcal{R}_\phi^\beta(\mathcal{X}, h) \leq \mathcal{R}_\phi(S, h)$) is to force the following constraint: $\sum_{i=1}^{m} \beta_i^{\text{-}y_i} y_i h(x_i) = 0$.

Unfortunately, the constraint $\sum_{i=1}^{m} \beta_i^{\text{-}y_i} y_i h(x_i) = 0$ still depends on the vector y of labels, which is not always provided and most likely uncertain or inaccurate in a weakly labeled data setting. We will show in Section A.3 that this issue can be overcome by means of an iterative 2-step learning procedure, that first learns a classifier minimizing the β-risk , possibly violating the constraint, and then learns a new matrix β that fulfills the constraint.

A.2.3 Instantiating β for Different Weakly Supervised Settings

The β-risk can be used as the basis for handling different learning settings, including weakly labeled learning. This can be achieved by fixing the β values, choosing their initial values or putting a prior on them. We have already seen that, fully supervised learning can be obtained by fixing all β values to 1 for the assigned class and to 0 for the opposite class. The current section provides guidance on how β could be instantiated to handle various weakly labeled settings.

In a *semi-supervised* setting, as detailed in the experimental section, we propose to initialize the β of unlabeled points to 0.5 and then to automatically refine them in an iterative process. Going further, and if we are ready to integrate spatial or topological information in the process, the β values of each unlabeled point could be initialized using a density estimation procedure (e.g., by considering the label proportions of the k nearest labeled neighbors). In the context of *multi-expert learning*, the experts' votes for each instance i can simply be averaged to produce the β_i values (or their initialization, or a prior). The case of *learning with label proportions* is especially useful for privacy-preserving data processing: the training points are grouped into bags and, for each bag, the proportion of labels are given. One way of handling such supervision is to initialize, for each bag, all the β with the same value that corresponds to the provided proportion of labels. *Noise-tolerant learning* aims at learning in the presence of label noise, where labels are given but can be wrong. For any point that can be possibly noisy, a direct approach is to use lower β values (instead of 1 in the supervised case) and refine them as in the semi-supervised setting. β can also be initialized using the label proportion of the k nearest labeled example (as done in the experimental section). The case of *Multiple Instance Learning* (MIL) is trickier: in a typical MIL setting, instances are grouped in bags and the supervision is given as a single label per bag that is positive if the bag contains at least one positive instance (negative bags contain only negative instances). A straightforward solution would be to recast the MIL supervision as a "learning with label proportion" (e.g., considering exactly one positive instance in each bag). It is not fully satisfying and a more promising solution would be to consider, within each bag, the set of β^{+1} variables and put a sparsity-inducing prior on them. This approach would be a less-constrained version of the relaxation proposed in WellSVM [118] (where it is supposed that there is exactly one positive instance per positive bag) and could be achieved by a l_1 penalty or using a Dirichlet prior (with low α to promote sparsity).

A.3 An iterative algorithm for weakly-labeled learning

As explained in Section A.2, a sufficient condition for guaranteeing that the β-risk is a convex upper-bound of the 0-1 loss based risk and it is not worse than the traditional risk is to fix $\sum_{i=1}^{m} \beta_i^{-y_i} y_i h(x_i) = 0$. However, the previous constraint depends on the labels. We overcome this problem by (i) iteratively learning a classifier minimizing the β-risk and most likely violating the constraint and then (ii) learning a new matrix β that fulfills it. The algorithm is generic. It can be used in different weakly labeled settings and can be instantiated with different losses and regularizations, as we will do in the next Section with SVMs.

As the process is iterative, let ${}^{t}\beta$ be the estimation of β at iteration t. At each iteration, our algorithm consists in two steps. We first learn an hypothesis h for the following problem P_1:

$$h^{t+1} = P_1(\mathcal{X}, {}^t\beta) = \arg\min_h \, c\mathcal{R}_\phi^{{}^t\beta}(\mathcal{X}, h) + \mathcal{N}(h)$$

which boils down to minimizing the $\mathcal{N}$-regularized empirical surrogate β-risk over the training sample $\mathcal{X}$ of size m, where $\mathcal{N}$, for instance, can take the form of a L_1 or a L_2 norm.

Then, we find the optimal β of the following problem P_2 for the points of $\mathcal{X}$:

$$^{t+1}\beta = P_2(\mathcal{X}, h^{t+1}) = \arg\min_\beta \mathcal{R}_\phi^\beta(\mathcal{X}, h^{t+1})$$

$$s.t. \sum_{i=1}^m \beta_i^{\text{-}y_i}(-y_i \, h^{t+1}(x_i)) = 0$$

$$\beta_i^{\text{-}1} + \beta_i^{+1} = 1, \;\; \beta_i^{\text{-}1} \geq 0, \beta_i^{+1} \geq 0 \;\; \forall i = 1..m \, .$$

For this step, a vector of labels is required. We choose to re-estimate it at each iteration according to the current value of β: we affect to an instance the most probable label, i.e. the σ corresponding to the biggest β^σ. The matrix β has to be initialized at the beginning of the algorithm according to the problem setting (see Section A.2.3). While some stabilization criterion does not exceed a given threshold ϵ, the two steps are repeated.

A.4 Soft-margin β-SVM

A major advantage of the empirical surrogate β-risk is that it can be plugged in numerous learning settings without radically modifying the original formulations. As an example, in this section we derive a new version of the Support Vector Machine problem, using the empirical surrogate β-risk , that takes into account the knowledge provided for each training instance (through the matrix β).

The soft-margin β-SVM optimization problem is a direct generalization of a standard soft-margin SVM and is defined as follows:

$$\arg\min_\theta \frac{1}{2} \|\theta\|_2^2 + c \sum_{i=1}^m \left(\beta_i^{\text{-}1} \xi_i^{\text{-}1} + \beta_i^{+1} \xi_i^{+1} \right)$$

$$s.t. \, \sigma(\theta^T \mu(x_i) + b) \geq 1 - \xi_i^\sigma \;\; \forall i = 1..m, \sigma \in \{-1, 1\}$$

$$\xi_i^\sigma \geq 0 \;\; \forall i = 1..m, \sigma \in \{-1, 1\}$$

where $\theta \in X'$ is the vector defining the margin hyperplane and b its offset, $\mu : X \to X'$ a mapping function and $c \in \mathbb{R}$ a tuned hyper-parameter. In the rest of the paper, we will refer to $K : X \times X \to \mathbb{R}$ as the kernel function corresponding to μ, i.e. $K(x_i, x_j) = \mu(x_i)\mu(x_j)$.

The corresponding Lagrangian dual problem is given by (the complete derivation is provided in the supplementary material):

$$\max_{\alpha} \ -\frac{1}{2}\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \text{\{-1,+1\}}}}\sum_{j=1}^{m}\sum_{\substack{\sigma'\in\\ \text{\{-1,+1\}}}} \alpha_i^{\sigma}\sigma\alpha_j^{\sigma}\sigma' K(x_i,x_j)+\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \text{\{-1,+1\}}}} \alpha_i^{\sigma}$$

$$s.t.\ 0 \leq \alpha_i^{\sigma} \leq c\beta_i^{\sigma}\ \forall i = 1..m,\ \sigma \in \{-1,1\}$$

$$\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \text{\{-1,+1\}}}} \alpha_i^{\sigma}\sigma = 0\ \forall i = 1..m,\ \sigma \in \{-1,1\}$$

which is concave w.r.t. α as for the standard SVM.

The β-SVM formulation differs from the SVM one in two points: first, the number of Lagrangian multipliers is doubled, because we consider both positive and negative labels for each instance; second, the upper-bounds for α are not the same for all instances but depend on the given matrix β. Like the coefficient c in the classical formulation of SVM, those upper-bounds play the role of trade-off between under-fitting and over-fitting: the smaller they are, the more robust to outliers the learner is but the less it adapts to the data. It is then logical that the upper-bound for an instance i depends on β_i^{σ} because it reflects the reliability on the label σ for that instance: if the label σ is unlikely, its corresponding α_i^{σ} will be constrained to be null (and its adversary will have more chance to be selected as a support vector, as $\beta_i^{\sigma} + \beta_i^{-\sigma} = 1$). Also, those points for which no label is more probable than the other ($\beta_i^{\sigma} \to 0.5$) will have less importance in the learning process compared to those for which a label is almost certain. In order to fully exploit the advantages of our formulation, c has to be finite and bigger than 0. As a matter of fact, when $c \to \infty$ or $c \to 0$, the constraints become exactly those of the original formulation.

A.5 Experimental results

In the first part of this section, we present some experimental results obtained by adapting the iterative algorithm presented in Section A.3 for semi-supervised learning and combining it with the previously derived β-SVM . Note that some approaches based on SVMs have been already presented in the literature to address the problem of semi-supervised learning. Among them, TransductiveSVM [35] iteratively learns a separator with the labeled instances, classifies a subset of the unlabeled instances and adds it to the training set. On the other hand, WellSVM [118] combines the classical SVM with a label generation strategy that allows one to learn the optimal separator, even when the training sample is not completely labeled, by convexly relaxing the original Mixed-Integer Programming problem. In [118], WellSVM has been shown to be very effective and better than TransductiveSVM and the state of the art. For this reason, we compare in this section β-SVM to WellSVM. In the second subsection, we present some preliminary results in the noise-tolerant learning setting, showing how β-SVM behaves when facing data with label noise.

A.5.1 Iterative β-SVM for Semi-Supervised Learning

We compare our method's performances to those of WellSVM, that has been proved, in [118], to performs in average better than the state of the art semi-supervised learning methods based on SVM and the standard SVM as well. In a semi-supervised context, a set $\mathcal{X}_l$ of labeled instances of size m_l and a set $\mathcal{X}_u$ of unlabeled instances of size m_u are provided. The matrix β is initialized as follows:

$$\forall i = 1..m_l \,\text{and}\, \forall \sigma \,\text{in}\, \{-1, 1\},\ {}^0\beta_i^\sigma = 1 \,\text{if}\, \sigma = y_i, 0\,\text{otherwise},$$

$$\forall i = m_l{+}1..m_u \,\text{and}\, \forall \sigma \,\text{in}\, \{-1, 1\},\ {}^0\beta_i^\sigma = 0.5$$

and we learn an optimal separator:

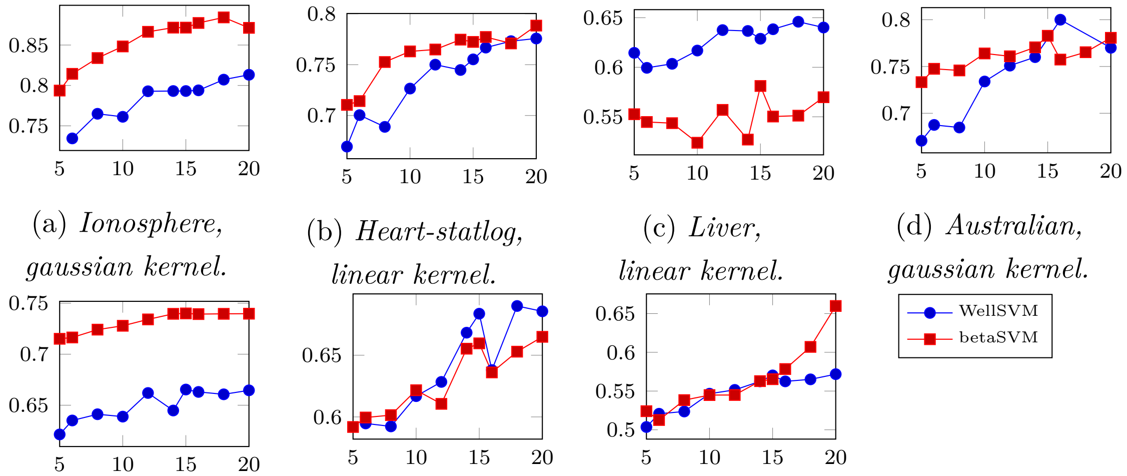
$$h^{t+1} = P_1(\mathcal{X}_l \cup \mathcal{X}_u, {}^t\beta) = \arg\min_h \, c_1 \mathcal{R}_\phi^{{}^t\beta}(\mathcal{X}_l, h) + c_2 \mathcal{R}_\phi^{{}^t\beta}(\mathcal{X}_u, h) + \mathcal{N}(h).$$

Here c_1 and c_2 are balance constants between the labeled and unlabeled set: when the number of unlabeled instances become greater than the number of labeled instances, we need to reduce the importance of the unlabeled set in the learning procedure because there exists the risk that the labeled set will be ignored. We consider the provided labels to be correct, so we keep the corresponding ${}_l\beta$ fixed during the iterations of the algorithm and estimate ${}_u\beta$ by optimizing $P_2(\mathcal{X}_u, h^{t+1})$. The iterative algorithm with β-SVM is implemented in Python using Cvxopt (for optimizing β-SVM) and Cvxpy [1] with its Ecos solver [55].

For each dataset, we show in Figure A.1 the accuracy of the two methods with an increasing proportion of labeled data. The different approaches are compared on the same kernel, either the linear or the gaussian, the one that gives higher overall accuracy. As a matter of fact, the choice of the kernel depends on the geometry of the data, not on the learning method.

For each proportion of labeled data, we perform a 4-fold cross-validation and we show the average accuracy over 10 iterations. Concerning the hyper-parameters of the different methods, we fix c_2 of β-SVM to $c_1\frac{m_l}{m}$, c_1 of WellSVM to 1 as explained in [118] and all the other hyper-parameters (c_1 for β-SVM and c_2 for WellSVM) are tuned by cross-validation through grid search. As for the stopping criteria, we fix ϵ of β-SVM to $10^{-5} + 10^{-3}\|h\|_{\mathcal{F}}$ and ϵ of WellSVM to 10^{-3} and the maximal number

[1] http://cvxopt.org/ and http://www.cvxpy.org/



(a) *Ionosphere, gaussian kernel.*

(b) *Heart-statlog, linear kernel.*

(c) *Liver, linear kernel.*

(d) *Australian, gaussian kernel.*

(e) *Pima, linear kernel.*

(f) *Sonar, linear kernel.*

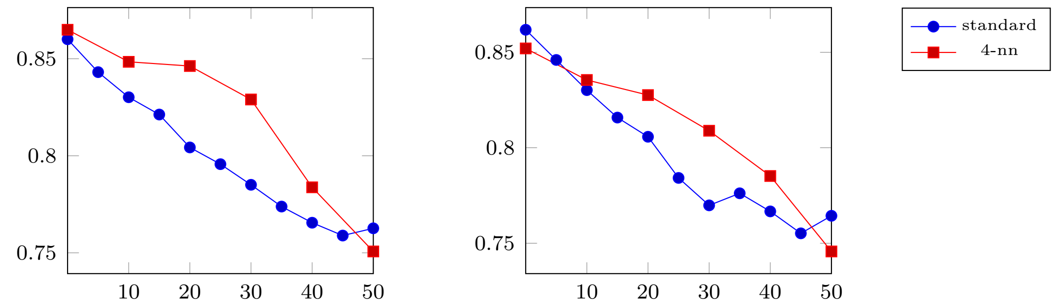
(g) *Splice, gaussian kernel.*

Figure A.1 – *Comparison of the mean accuracies of WellSVM and β-SVM versus the percentage of labeled data on different UCI datasets.*

of iterations to 20 for both methods. When using the gaussian kernel, the γ in $K(x_i, x_j) = \exp(-\|x_i - x_j\|_2^2/\gamma)$ is fixed to the mean distance between instances. Our method performs better than WellSVM, with few exceptions, and is more efficient in terms of CPU time: for the Australian dataset, the biggest dataset in number of features and instances, WellSVM is in average 30 times slower than our algorithm (without particular optimization efforts).

A.5.2 Preliminary Results Under Label-Noise

We quickly tackle another setting of the weakly labeled data field: the noise-tolerant learning, the task of learning from data that have noisy or uncertain labels. It has been shown in [28] that SVM learning is extremely sensitive to outliers, especially the ones lying next to the boundary. We study, the sensitivity of β-SVM to label noise artificially introduced on the Ionosphere dataset. We consider two initialization strategies for β: the *standard* on where $\beta^{y_i} = 1$ and $\beta^{-y_i} = 0$ and the *4-nn* one where β^{σ} is set to the proportion of neighboring instances with label σ. In Figure A.4, we draw the mean accuracy over 4 repetitions w.r.t. an increasing percentage (as a proportion of the smallest dataset) of two kinds of noise: the symmetric noise, introduced by swapping the labels of instances belonging to different classes, and the asymmetric noise, introduced by gradually changing the labels of the instances of one class. These preliminary results are encouraging and show that locally estimating the conditional class density to initialize the β matrix improves the robustness of our method to label noise.



(a) *Symmetric Noise.* (b) *Asymmetric Noise.*

Figure A.2 – *Comparison of the mean accuracy versus the percentage of noise of iterative β-SVM with different initializations of β. The standard curve refers to the initialization of $\beta^{y_i} = 1$ and $\beta^{-y_i} = 0$ and the 4-nn to the initialization of β^{σ} to the proportion of neighboring instances with label σ.*

A.6 Conclusion

This paper focuses on the problem of learning from weakly labeled data. We introduced the β-risk which generalizes the standard empirical risk while allowing the integration of weak supervision. From the expression of the β-risk , we derived a generic algorithm for weakly labeled data and specialized it in an SVM-like context. The resulting β-SVM algorithm has been applied in two different weakly labeled settings, namely semi-supervised learning and learning with label noise, showing the advantages of the approach.

The perspectives of this work are numerous and of two main kinds: covering new weakly labeled settings and studying theoretical guarantees. As proposed in Section A.2.3, the β-risk can be used in various weakly labeled scenarios. This requires to use different strategies for the initialization and the refinement of β, and also to propose proper priors for these parameters. Generalizing the proposed β-risk to a multi-class setting is a natural extension as β is already a matrix of class probabilities. Another broad direction involves deriving robustness and convergence bounds for the algorithms built on the β-risk .

A.7 Acknowledgments

We thank the reviewers for their valuable remarks. We also thank the ANR projects SOLSTICE (ANR-13-BS02-01) and LIVES (ANR-15-CE230026-03).

A.8 Supplementary material

A.8.1 Overview

This supplementary material is organized as follows: in Section A.8.2, we prove the property of a Legendre conjugate of a permissible function used in Eq.(2) of Sec.(2) of the paper; in Section A.8.3, we derive the dual problem of a soft-margin β-SVM ;

A.8.2 Legendre Conjugate of Permissible Functions

The Legendre conjugate of a differentiable and strictly convex function ϕ can be written as:

$$\phi^*(x) = x\nabla_\phi^{-1}(x) - \phi(\nabla_\phi^{-1}(x)).$$

In the case of a permissible function ϕ, its Legendre conjugate has the following property: $\phi^*(-x) = \phi^*(x) - x$.

Proof.

$$\begin{aligned}
\phi^*(-x) &= -x\nabla_\phi^{-1}(-x) - \phi(\nabla_\phi^{-1}(-x)) \\
&= -x(1 - \nabla_\phi^{-1}(x)) - \phi(1 - \nabla_\phi^{-1}(x)) \qquad \text{(A.6)} \\
&= -x + x\nabla_\phi^{-1}(x) - \phi(\nabla_\phi^{-1}(x)) \qquad \text{(A.7)} \\
&= \phi^*(x) - x
\end{aligned}$$

Because of the symmetry of ϕ about $-\frac{1}{2}$, in Eq. (A.6) $\nabla_\phi^{-1}(-x) = 1 - \nabla_\phi^{-1}(x)$ and in Eq. (A.7) $\phi(1-x) = \phi(x)$. □

A.8.3 Derivation of Soft-margin β-SVM

The soft-margin β-SVM optimization problem is a direct generalization of a standard soft-margin SVM and is defined as follows:

$$\arg\min_\theta \frac{1}{2}\|\theta\|_2^2 + c\sum_{i=1}^{m}\left(\beta_i^{-1}\xi_i^{-1} + \beta_i^{+1}\xi_i^{+1}\right)$$

$$s.t.\, \sigma(\theta^T\mu(x_i) + b) \geq 1 - \xi_i^\sigma \;\forall i = 1..m, \sigma \in \{-1, 1\}$$

$$\xi_i^\sigma \geq 0 \;\forall i = 1..m, \sigma \in \{-1, 1\}$$

where $\theta \in X'$ is the vector defining the margin hyperplane and b its offset, $\mu : X \to X'$ a mapping function and $c \in \mathbb{R}$ a tuned hyper-parameter. In the rest of the paper, we will refer to $K : X^2 \to \mathbb{R}$ as the kernel function corresponding to μ $(K(x_i, x_j) = \mu(x_i)\mu(x_j))$.

Instead of solving the previous primal problem, it is preferable to solve its Lagrangian dual problem given by maximizing the corresponding Lagrangian w.r.t. its Lagrangian multipliers, which gives a nice Quadratic Programming problem that can be solved by common optimization techniques. The Lagrangian can be written as follows:

$$\begin{aligned}\mathcal{L}(\theta, b, \xi, \alpha, r) =& \frac{1}{2}\|\theta\|_2^2 + c\sum_{i=1}^{m}\left(\beta_i^{-1}\xi_i^{-1} + \beta_i^{+1}\xi_i^{+1}\right)\\ &-\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{-1,+1\}}}\alpha_i^\sigma\left(\sigma(\theta^T\mu(x_i)+b)+\xi_i^\sigma-1\right)-\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{-1,+1\}}} r_i^\sigma\xi_i^\sigma\end{aligned} \tag{A.8}$$

where $\alpha \in \mathbb{R}^{2*m}$ and $r \in \mathbb{R}^{2*m}$ are the Lagrangian multipliers. It is obvious that:

$$\max_{\alpha,r\geq 0}\min_{\theta,b,\xi}\mathcal{L}(\theta, b, \xi, \alpha, r) \leq \min_{\theta,b,\xi}\max_{\alpha,r\geq 0}\mathcal{L}(\theta, b, \xi, \alpha, r)$$

where the left term corresponds to the optimal value of the dual problem and the right one to the primal's one. The dual and the primal problems have the same value at optimality if the Karush-Kuhn-Tucker (KKT) conditions are not violated (see [32]). By setting the gradient of $\mathcal{L}$ w.r.t. θ, b and ξ to 0, we find the saddle point corresponding to the function minimum:

$$\nabla_\theta\mathcal{L}(\theta, b, \xi, \alpha, r) = \theta - \sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{-1,+1\}}}\alpha_i^\sigma\sigma\mu(x_i)$$

$$\nabla_b\mathcal{L}(\theta, b, \xi, \alpha, r) = -\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{-1,+1\}}}\alpha_i^\sigma\sigma$$

$$\nabla_{\xi_i^\sigma}\mathcal{L}(\theta, b, \xi, \alpha, r) = c\beta_i^\sigma - \alpha_i^\sigma - r_i^\sigma$$

which give

$$\theta = \sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{-1,+1\}}}\alpha_i^\sigma\sigma\mu(x_i) \tag{A.9}$$

$$\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{-1,+1\}}}\alpha_i^\sigma\sigma = 0 \tag{A.10}$$

$$\alpha_i^\sigma \leq c\beta_i^\sigma \tag{A.11}$$

We can now write the QP dual problem by replacing θ by its expression (E.1) and simplifying following (E.2) and (E.3):

$$\max_{\alpha} \; -\frac{1}{2}\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_i^\sigma \sigma \sum_{j=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_j^\sigma \sigma K(x_i,x_j) + \sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_i^\sigma$$

$$s.t.\; 0 \leq \alpha_i^\sigma \leq c\beta_i^\sigma \;\forall i = 1..m,\, \sigma \in \{-1,1\}$$
$$\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_i^\sigma \sigma = 0 \;\forall i = 1..m,\, \sigma \in \{-1,1\}$$

which is concave w.r.t. α.

Proof.

$$\begin{aligned}\mathcal{L}(\alpha) = {} & \frac{1}{2}\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_i^\sigma \sigma \mu(x_i) \sum_{j=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_j^\sigma \sigma \mu(x_j) + c\sum_{i=1}^{m}\left(\beta_i^{\text{-1}}\xi_i^{\text{-1}} + \beta_i^{+1}\xi_i^{+1}\right) \\ & - \sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_i^\sigma \left(\sigma\left(\left(\sum_{j=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_j^\sigma \sigma \mu(x_j)\right)\mu(x_i) + b\right) + \xi_i^\sigma - 1\right) - \sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} r_i^\sigma \xi_i^\sigma \end{aligned} \tag{A.12}$$

$$= -\frac{1}{2}\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_i^\sigma \sigma \mu(x_i) \sum_{j=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_j^\sigma \sigma \mu(x_j) + \sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_i^\sigma$$
$$+ \sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \left(c\beta_i^\sigma - \alpha_i^\sigma - r_i^\sigma\right)\xi_i^\sigma - b\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_i^\sigma \sigma \tag{A.13}$$
$$= -\frac{1}{2}\sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_i^\sigma \sigma \sum_{j=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_j^\sigma \sigma K(x_i,x_j) + \sum_{i=1}^{m}\sum_{\substack{\sigma\in\\ \{\text{-1,+1}\}}} \alpha_i^\sigma \tag{A.14}$$

In Eq. (A.13) the third and the fourth terms are null because of (E.2) and (E.3). □

We need the following two additional constraints in order to respect the KKT conditions which justify guarantee that the optimal value found by solving the dual problem corresponds to the optimal value of the primal:

$$\alpha_i^\sigma\left(\sigma(\theta^T + b) - 1 + \xi_i^\sigma\right) = 0 \,\forall\, i = 1..m, \sigma \in \{-1,1\}$$

$$r_i^\sigma \xi_i^\sigma = 0 \,\forall\, i = 1..m, \sigma \in \{-1,1\}$$

Once the Lagrangian dual problem solved, the characteristic vector θ and offset b of the optimal margin hyperplane can be retrieved by means of the support vectors machine, i.e. the instances whose corresponding α_i^σ are strictly greater than 0:

$$\theta = \sum_{i)1}^{m} \sum_{\substack{\sigma \in \\ \{\text{-1,+1}\}}} \alpha_i^\sigma \sigma \mu(x_i)$$

$$b = \theta\mu(x_k) - \sigma_k$$

and the new instances can be classified :

$$y(x) = sign(\sum_{i=1}^{m} \sum_{\substack{\sigma \in \\ \{\text{-1,+1}\}}} (\alpha_i^\sigma \sigma K(x_i, x)) + b)$$

A.8.4 Additional Experiments: Semi-Supervised Learning

We report a table of mean accuracies with their relative errors of the performances of standard SVM, WellSVM and our method on 7 UCI datasets with 5%,10% and 15% of labeled instances of the training sets.

A.8.5 Additional Experiments: Robustness to Label Noise

Here we report the results of applying β-SVM to a synthetic dataset and study its robustness to artificially induced label noise.

The synthetic dataset consists in 40 instances of 2 balanced classes: the instances of each class are uniformly distributed around a center point so that they can be easily classified by a linear separator to which we will refer as the true separator.

In Fig. A.4, we compare the linear classifiers learned at each iteration of our iterative, algorithm with β-SVM , with a standard linear SVM and with the true separator. We conducted the experiment as follows: we apply the two methods first on the original dataset, then on a dataset where we swapped the label of a random instance of each class and so on with an increasing number of swapped labels.

We notice that our method is more robust to label noise: even though at the first iteration, we learn the same separator as the standard linear SVM, through the following iterations the algorithm converges to a separator closer to the true separator.

Table A.1 – *Mean accuracies with their relative errors of the performances of standard SVM, WellSVM and our method on 7 UCI datasets with 5%,10% and 15% of labeled instances of the training sets.*

Dataset	% labeled	SVMs	WellSVM	betaSVM
ionosphere	5	0.74 ± 0.02	0.72±0.04	**0.77±0.03**
	10	0.78 ± 0.03	0.79±0.03	**0.80±0.04**
	15	0.81 ± 0.01	**0.82±0.02**	0.81±0.02
sonar	5	0.58 ± 0.06	0.58±0.03	**0.59±0.05**
	10	0.65 ± 0.04	0.64±0.04	**0.66±0.05**
	15	0.65 ± 0.02	**0.67±0.02**	**0.67±0.02**
liver	5	0.59±0.02	**0.61±0.04**	0.55±0.04
	10	0.61±0.04	**0.64±0.03**	0.58±0.03
	15	**0.64±0.04**	**0.64±0.03**	0.58±0.03
splice	5	**0.53±0.07**	0.50±0.07	**0.53±0.06**
	10	**0.56±0.02**	0.55±0.05	0.55±0.07
	15	**0.60±0.03**	0.56±0.05	0.56±0.04
heart-statlog	5	0.64±0.04	0.55±0.03	**0.71±0.04**
	10	0.72±0.03	0.62±0.02	**0.76±0.03**
	15	0.73±0.02	0.63±0.03	**0.77±0.02**
australian	5	0.72 ± 0.05	0.64 ± 0.01	**0.73±0.06**
	10	**0.73±0.03**	0.72 ± 0.04	**0.73±0.04**
	15	**0.76±0.07**	0.75 ± 0.03	0.75 ± 0.04
pima	5	0.65±0.01	0.62±0.03	**0.71±0.01**
	10	0.69±0.01	0.63±0.03	**0.72±0.01**
	15	0.71±0.01	0.64±0.03	**0.72±0.01**

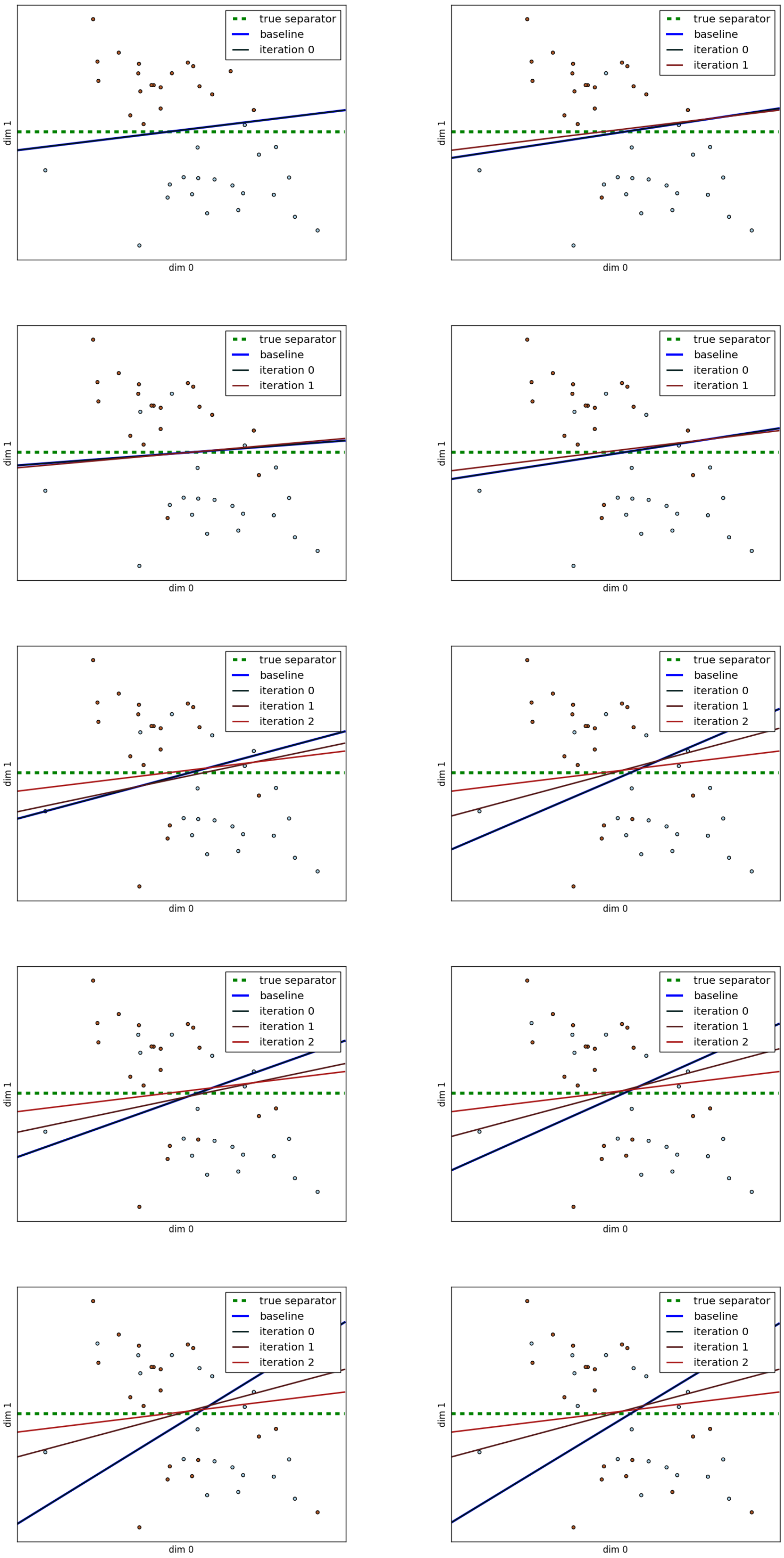


Figure A.3 – *Artificially induced label noise: the baseline, here, corresponds to the separator learned with a classical SVM. The first figure shows the learned separators with the original labels, and the other figures show the results for an increasing number of swapped labels going from left to right and from to bottom.*

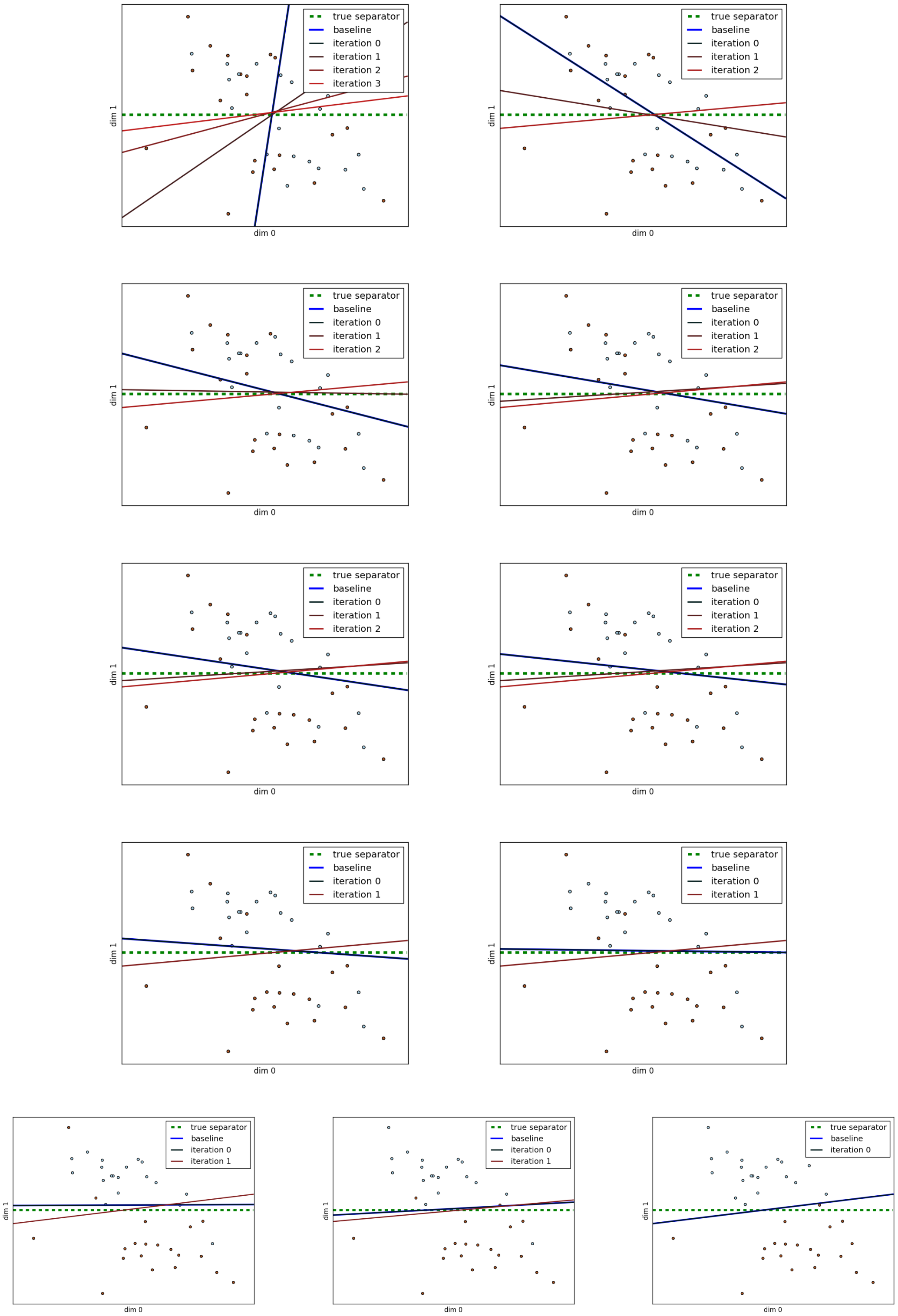


Figure A.4 – *Artificially induced label noise: the baseline, here, corresponds to the separator learned with a classical SVM. The first figure shows the learned separators with the original labels, and the other figures show the results for an increasing number of swapped labels going from left to right and from to bottom.*

Efficient Defenses Against Adversarial Attacks

B

This Appendix reports the publication

Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 39–49. ACM, 2017.

Following the recent adoption of deep neural networks (DNN) accross a wide range of applications, adversarial attacks against these models have proven to be an indisputable threat. Adversarial samples are crafted with a deliberate intention of undermining a system. In the case of DNNs, the lack of better understanding of their working has prevented the development of efficient defenses. In this paper, we propose a new defense method based on practical observations which is easy to integrate into models and performs better than state-of-the-art defenses. Our proposed solution is meant to reinforce the structure of a DNN, making its prediction more stable and less likely to be fooled by adversarial samples. We conduct an extensive experimental study proving the efficiency of our method against multiple attacks, comparing it to numerous defenses, both in white-box and black-box setups. Additionally, the implementation of our method brings almost no overhead to the training procedure, while maintaining the prediction performance of the original model on clean samples.

B.1 Introduction

Deep learning has proven its prowess across a wide range of computer vision applications, from visual recognition to image generation [113]. Their rapid deployment in critical systems, like medical imaging, surveillance systems or security-sensitive applications, mandates that reliability and security are established *a priori* for deep learning models. Similarly to any computer-based system, deep learning models can

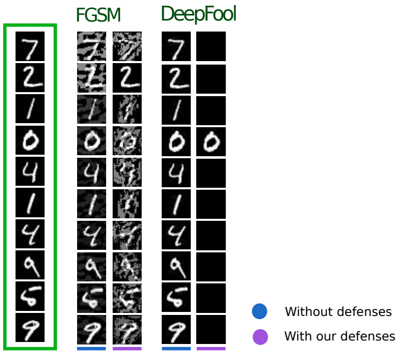


Figure B.1 – *Minimal perturbations needed for fooling a model on the first ten images from MNIST. The original examples are marked by the green rectangle. With our defenses, the attack becomes visually detectable.*

potentially be attacked using all the standard methods (such as denial of service or spoofing attacks), and their protection only depends on the security measures deployed around the system. Additionally, DNNs have been shown to be sensitive to a threat specific to prediction models: *adversarial examples* [22, 179]. These are input samples which have deliberately been modified to produce a desired response by a model (often, misclassification or a specific incorrect prediction which would benefit the attacker).

Adversarial examples pose an asymmetric challenge with respect to attackers and defenders. An attacker aims to obtain his reward from a successful attack without raising suspicion. A defender on the other hand is driven towards developing strategies which can guard their models against all known attacks and ideally for all possible inputs. Furthermore, if one would try to prove that a model is indeed secure (that is, can withstand attacks yet to be designed), one would have to provide formal proof, say, through verification [86]. This is a hard problem, as this type of methods does not scale to the number of parameters of a DNN. For all these reasons, finding defense strategies is hard.

Apart from the security aspects, adversarial examples for image classification have other peculiar properties. First and foremost, the imperceptible difference between adversarial and legitimate examples provides them with an effortless capacity of attack; this is only reinforced by the transferability of such samples across different models, allowing for black-box attacks [22, 149]. Moreover, the high confidence of misclassification proves that the model views adversarial samples as regular inputs. The potential damage of adversarial attacks is increased by the existence of such samples in the physical world [103]: showing printed versions of adversarial images to a camera which will feed then to a DNN has the same effect as just presenting them to the model without passing through a physical support. Another intriguing property is that nonsensical inputs (e.g. crafted noise) are interpreted by the model as natural examples with high confidence [20, 143]. From the perspective of learning, these counterintuitive aspects are related to the more fundamental questions of what does a model learn and how well it is able to generalize [128, 223]. As a matter of fact, the mere existence of adversarial samples suggests that DNNs might be failing to extract concepts from the training set and that they would, instead, memorize it.

A good understanding of the weaknesses of deep learning models should bring to better attack strategies, and most importantly more effective defenses. While the cause is not completely understood to this day, multiple hypotheses have been suggested to explain their sensitivity to adversarial inputs. One of the first such hypothesis stated that the high complexity and non-linearity [179] of neural networks can assign random labels in areas of the space which are under-explored. Moreover, these adversarial examples would be rare and would only exist in small regions of the space, similar to pockets. These theories have been refuted, since they are unable justify the transferability of adversarial samples from one model to another. Moreover, it has been shown that linear models also suffer from this phenomenon. [77] proposed a *linearity hypothesis* instead: deep neural networks are highly non-linear with respect to their parameters, but mostly linear with respect to their inputs, and adversarial examples are easy to encounter when exploring a direction orthogonal to a decision boundary. Another conjecture to explain the existence of adversarial examples is the cumulation of errors while propagating the perturbations from layer to layer. A small carefully crafted perturbation in the input may result in a much greater difference in the output layer, effect that is only magnified in high dimensional spaces, causing the activation of the wrong units in the upper layers.

In this paper, we make the following contributions:

- We propose a two-fold defense method which is easy to setup and comes at almost no additional cost with respect to a standard training procedure. The method is designed to reinforce common weak points of deep networks and to smooth the decision functions. As a consequence, it is agnostic to the type of attack used to craft adversarial examples, making it effective in multiple settings. Figure B.1 shows examples of clean images which are then perturbed with standard attacks. When the model uses the proposed defense, the perturbation necessary for misclassification is much larger, making the attack detectable and, in some cases, turning the images into nonsense.

- We perform an extended experimental study, opposing an important number of attacks to the most effective defense methods available, alongside our proposed defense. We evaluate them according to multiple metrics and prove that accuracy by itself is not a sufficient score. We explore white-box and black-box attacks alike, to account for different adversarial capacities, as well as transferability of examples.

The rest of this document is structured as follows. Section B.2 provides an overview of the existing attack and defense methods. Section B.3 introduces the proposed defense method, followed by an extensive experimental study in Section B.4. We conclude this paper in Section B.5.

B.2 Related work

Background and notation A neural network can be formalized as a function $F(x, \theta) = y$ taking inputs $x \in \mathbb{R}^n$ and providing outputs $y \in \mathbb{R}^m$ w.r.t. a set of parameters θ. This paper covers the case of multi-class classification, where the last layer in the network is the *softmax* function, and m is the number of labels. The *softmax* function has two main properties: (i) it amplifies high values, while diminishing smaller ones, and (ii) it outputs vectors y of non-negative values which sum to 1, giving them an interpretation of probability distributions. The input x is then attributed the class label with the highest probability. Now consider a network F containing L layers, F_1 being the input and F_L the *softmax* layer. An internal layer F_j can be written as:

$$F_j(x_{j-1}, (\theta_j, \hat{\theta}_j)) = \sigma(\theta_j \cdot x_{j-1}) + \hat{\theta}_j,$$

where $\sigma(\cdot)$ is the activation function, θ_j and $\hat{\theta}_j$ are the parameters of layer F_j and x_{j-1} is the output of the previous layer. The most popular activation function, and almost the only one that has been studied in an adversarial setting, is the Rectified Linear Unit, or RELU [74].

In our study of adversarial learning, we focus on the task of image classification, as this type of data is readily interpretable by humans: it is possible to distinguish true adversarial examples, i.e. perceptually identical to the original points, from rubbish ones, i.e. overly perturbed and meaningless. Consider an image of size $w \times h$ pixels with pixel values scaled between 0 and 1. In the greyscale case, such an image can be viewed as a vector $x \in \mathbb{R}^{w \cdot h}$, where each x_i denotes a pixel. Similarly, RGB images have three color channels and are written as vectors $x \in \mathbb{R}^{3 \cdot w \cdot h}$. The problem of crafting adversarial examples can be formulated as trying to find samples $x' = x + \Delta x$ which fool the model into making incorrect predictions. Adversarial machine learning is formalized for the first time in the pioneering work of [51] and [126], where game theoretical approaches are proposed to attack (linear) classification models. [11] provide an extensive taxonomy of attack types against machine learning, encompassing both the attacks on test sets considered in the present paper (also known as *evasion attacks*), as well as training set tampering (*poisoning attacks*). The purpose of their proposed framework, as well as that of following work from [24], is to provide the tools for determining under which considerations machine learning can be secure.

[179] and [22] are the first adversarial attacks against deep neural networks, aiming to achieve a target class prediction. The method in [179] is expressed as a box-constrained optimization objective, solved through L-BFGS:

$$\min_{\Delta x} F(x + \Delta x) \neq F(x), \quad \text{s.t.} \quad x' \in [0,1]^n. \tag{B.1}$$

Although effective in producing adversarial examples, this attack is computationally expensive to the point where its usage is not practical. Its counterpart in [22] uses a similar objective, enhanced by a regularizer aiming to produce attack samples placed in high density input regions. This provides the attack with better mimicking and concealing capacitites. In order to speed up the computation of adversarial examples, the attack methods subsequently proposed [38, 77, 139, 151] all solve a first-order approximation of Problem (B.1), which has a geometrical interpretation. The resulting perturbations are also effective in fooling the model, probably because commonly used deep architectures, such as the ones with piecewise linear RELU, are highly linear w.r.t. the input [77].

We now discuss the methods which have shaped the current state-of-the-art in adversarial attacks for deep neural networks. Any of the following attacks can be deployed in two ways: either by crafting the adversarial examples having knowledge of the architecture and the parameters of the attacked model (white-box attacks) or by crafting them using a similar model, plausible for the considered task, or a surrogate one as in [149] without exploiting any sensitive information (black-box attacks).

Attacks One of the gradient-based methods is the Jacobian saliency map attack (JSMA) [151], which uses the derivative of the neural network with respect to the input image to compute the distortion iteratively. Each iteration, the pixel with the highest derivative is modified by a fixed value (the budget for the attack), followed by recomputing the saliency map, until the prediction has changed to a target class. The adversarial images produced by JSMA are subtle and effective for attacks, but they still require an excessive amount of time to compute.

The fast gradient sign method (FGSM) [77] has been introduced as a computationally inexpensive, but effective alternative to JSMA. FGSM explores the gradient direction of the cost function and introduces a fixed amount of perturbation to maximize that cost. In practice, the examples produced by this attack are more easily detectable and require a bigger distortion to achieve misclassification than those obtained from JSMA. An iterative version of FGSM, where a smaller perturbation is applied multiple times, was introduced by [103].

Instead of using a fixed attack budget as for the last two methods, DeepFool [139] was the first method to compute and apply the minimal perturbation necessary for misclassification under the L_2 norm. The method performs iterative steps on the adversarial direction of the gradient provided by a locally linear approximation of the classifier. Doing so, the approximation is more accurate than FGSM and faster than JSMA, as all the pixels are simultaneously modified at each step of the method, but its iterative nature makes DeepFool computationally expensive. In [137, 138], the authors extend DeepFool in order to craft a *universal perturbation* to be applied indifferently to any instance: a fixed distortion is computed from a set of inputs, allowing to maximize the predictive error of the model on that sample. The perturbation is computed by a greedy approach and needs multiple iterations over the given sample before converging. To the extent where the sample is representative to the data distribution, the computed perturbation has good chances of achieving misclassification on unseen samples as well.

One method aiming to compute good approximations of Problem (B.1) while keeping the computational cost of perturbing examples low has been proposed in [38]. The authors cast the formulation of [179] into a more efficient optimization problem, which allows them to craft effective adversarial samples with low distortion. They define three similar targeted attacks, based on different distortion measures: L_2, L_0 and L_∞ respectively. In practice, even these attacks are computationally expensive.

If it is difficult to find new methods that are both effective in jeopardizing a model and computationally affordable, defending from adversarial attacks is even a harder task. On one hand, a good defense should harden a model to any known attack and, on the other hand, it should not compromise the discriminatory power of the model. In the following paragraph, we report the most effective defenses proposed for tackling adversarial examples.

Defenses A common technique for defending a model from adversarial examples consists in augmenting the training data with perturbed examples (technique known as ‘adversarial training‘ [179]) by either feeding a model with both true and adversarial examples or learning it using the modified objective function:

$$\hat{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha) J(\theta, x + \Delta x, y)$$

with J the original loss function. The aim of such defense is to increase the model's robustness in specific directions (of the adversarial perturbation) by ensuring that it will predict the same class for the true example and its perturbations along those directions. In practice, the additional instances are crafted for the considered model using one or multiple attack strategies, such as FGSM [77], DeepFool [139] and virtual adversarial examples [136].

However, adversarially training a model is effective only on adversarial examples crafted on the original model, which is an improbable situation considering that an attacker might not have access to exactly the same model for computing the perturbations. Additionally, adversarial training has been proved to be easily bypassed through a two-step attack [185], which first applies a random perturbation to an instance and then performs any classical attack technique. The success of this new attack, and of black-box attacks in general, is due to the sharpness of the loss around the training examples: if smoothing the loss in few adversarial directions makes ineffective gradient-based attacks on those directions, it also makes the loss sharper in the other directions, leaving the model more vulnerable through new attacks.

Unlike adversarial training, a different family of defenses aims to increase the robustness of deep neural networks to adversarial examples independently of the attack. Among these *attack-agnostic* techniques, we find defensive distillation [36, 150, 152], which hardens the model in two steps: first, a classification model is trained and its *softmax* layer is smoothed by division with a constant T; then, a second model is trained using the same inputs, but instead of feeding it the original labels, the probability vectors from the last layer of the first model are used as soft targets. The second model is then used for future deployment. The advantage of training the second model with this strategy is that it makes for a smoother loss function. It has been shown in [201] that a similar behavior can be obtained at a cheaper cost by training a model using smoothed labels. This technique, called *label smoothing*, involves converting class labels into soft targets (value close to 1 for the target class and the rest of the weight distributed on the other classes) and use these new values for training the model instead of the true labels. As a consequence, one saves the needs to train an additional model as for defensive distillation.

Another model hardening technique is feature squeezing [210, 211]. Its reduces the complexity of the representation of the data so that the adversarial perturbations disappear due to low sensitivity. The authors propose two heuristics for dealing with images: reducing color depth on a pixel level, that is encoding the colors with less values, and using a smoothing filter over the image. As an effect, multiple inputs are mapped to the same value, making the model robust to noise and adversarial attacks. Although this has the collateral effect of worsening the accuracy of the model on true examples, to the best of our knowledge, feature squeezing is the most effective defense to adversarial attacks to date.

A different approach for protecting models against adversarial attacks are detection systems. To this end, a certain number of directions have been explored, such as performing statistical tests [79], using an additional model for detection [76, 132] or applying dropout [173] at test time [62]. However, with adversarial examples being relatively close to the original distribution of the data, it has been shown that many detection methods can be bypassed by attackers [37].

As we have described, defending against adversarial examples is not an easy task, and the existing defense methods are only able to increase model robustness in certain settings and to a limited extent. With these aspects in mind, we now move on to presenting the proposed defense method.

B.3 Efficient defenses

In this section, we start by introducing the threat model we consider, before presenting the two aspects of our contribution.

Adversary model Attackers can be formalized depending on their degree of knowledge, the ways in which they can tamper with the system, as well as the expected reward. For the purpose of our contribution, modeling the reward is not required. In this paper, we consider attackers that only have access to test data and, optionally, the trained model. They are thus unable to tamper with the training sample, unlike in other contexts, such as learning with malicious error [94]. We address different settings, depending on the degree of knowledge of the adversary. The attacker can gain access to information about the learning algorithm, which can include only the architecture of the system or values of the parameters as well, the feature space and the data which was used for training. Of course, from the perspective of the attacker, the white-box setup is the most advantageous, making the crafting easier. A good defense method should be able to sustain the strongest type of attack achievable in practice. On the other hand, it has been shown that in some cases a black-box attack, when the attacker only has access to the input and output of the model, achieves better results than its white-box counterpart [149]. We thus consider both black-box and white-box attacks when evaluating our method.

B.3.1 Bounded RELU

We now introduce the use of the bounded RELU (BRELU) [121] activation function for hedging against the forward propagation of adversarial perturbation. Activation functions present in each node of a deep neural network amplify or dampen a signal depending on the magnitude of the input received. Traditional image classification models use Rectified Linear Units which are known to learn sparse representations for data [74], thereby easing the training process. Recall that a RELU operation squashes negative inputs to zero, while propagating all positive signals. Given the arrangement of nodes in a DNN, inputs received by a node in one layer depend on the outputs of the nodes from the previous layers of the architecture. Naturally, with unbounded units like RELU, a small perturbation in the input signal can accumulate over layers as a signal propagates forward through the network. For an adversarial perturbation, this can potentially lead to a significant change in the output signal for an incorrect class label. This would be further amplified by the *softmax* operation in the final layer of a classification network. To curtail this phenomenon, we propose the use of the bounded RELU activation function, defined as follows:

$$f_t(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \leq x < t \\ t, & x \geq t. \end{cases}$$

The parameter t defines the cut-off point where the function saturates. In practice, it should be set with respect to the range of the inputs. Notice that a value too small for t would prevent the forward propagation of the input in the network, reducing the capacity of the model to perform well. On the other hand, too big a value will perpetuate the same behavior as RELU.

To theoretically prove the interest of this simple modification in the architecture, we compare the additive stabilities of a neural network with RELU activations against BRELU activations, following [179]. For a model using RELU, the output difference between an original point and its perturbation can be upper bounded as follows

$$\forall x, \Delta x \quad ||F(x) - F(x + \Delta x)|| \leq M||\Delta x||$$

with $M = \prod_{j=1}^{L} M_j$ the product of the Lipschitz constants of each layer. On the contrary, the output difference in a model with BRELU has a tighter bound for a small enough t and, most importantly, a bound independent from the learned parameters of the layers:

$$\forall x, \Delta x \quad ||F(x) - F(x + \Delta x)|| \leq t||\mathbf{1}||.$$

Employing BRELU activation functions, then, improves the stability of the network.

B.3.2 Gaussian Data Augmentation

The intuition behind data augmentation defenses such as adversarial training is that constraining the model to make the same prediction for a true instance and its slightly perturbed version should increase its generalization capabilities. Although adversarial training enhances the model's robustness to white-box attacks, it fails to protect effectively from black-box attacks [185]. This is because the model is strengthened only in few directions (usually, one per input sample), letting it be easily fooled in all the other directions. Moreover, there is no mechanism for preventing the model from making confident decisions for uncertain regions, i.e. parts of the input space not represented by the data samples. Instead, augmenting the training set with examples perturbed using Gaussian noise, as we propose in this paper, on one hand allows to explore multiple directions and, on the other, smooths the model confidence. While the former property can be achieved through any kind of noise (e.g. uniform noise), the latter is peculiar to using a Gaussian distribution for the perturbations: the model is encouraged to gradually decrease its confidence moving away from an input point.

We thus propose a new formulation of the problem of learning classifiers robust to adversarial examples:

$$\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathop{\mathbb{E}}_{\Delta x\sim\mathcal{N}(0,\sigma^2)} J(\theta, x+\Delta x, y),$$

where σ indicatively corresponds to the acceptable non-perceivable perturbation. The aim is to enforce the posterior distribution $p(y|x)$ to follow $\mathcal{N}(x,\sigma^2)$. This formulation differs from [128] in considering all possible local perturbations (and not only the one with the maximal threatening power) and in weighing them with respect to their magnitude.

In the rest of the paper, we will solve a Monte Carlo approximation of the solution of the previous problem, by sampling N perturbations per instance from $\mathcal{N}(0,\sigma^2)$:

$$\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \frac{1}{N}\sum_{i=1}^{N} J(\theta, x+\Delta x_i, y)$$

that converges almost surely to the true one for $N \to \infty$. Let's note $\mu = \mathbb{E}_{\Delta x\sim\mathcal{N}(0,\sigma^2)} J(\theta, x+\Delta x, y)$. After Hoeffding's inequality, the amount of deviation of the empirical approximation from the theoretical one can be quantified for all $t \geq 0$ as

$$\Pr\left[\left|\frac{1}{N}\sum_{i=1}^{N} J(\theta, x+\Delta x_i, y) - \mu\right| \geq t\right] \leq \exp\left(-\frac{t^2 N}{2\sigma^2}\right).$$

To prove the benefits of the proposed Gaussian data augmentation (**GDA**) on the robustness of a model, we carry out a study on the classification boundaries and the confidence levels of a simple multi-layer network trained on two toy datasets augmented through different techniques. In the first dataset (Figure B.2a), the two classes are placed as concentric circles, one class inside the other. The second dataset (Figure B.2b) is the classic example of two half-moons, each representing one class. For each original point x of a dataset, we add one of the following artificial points x' (all of them clipped into the input domain):

1. Adversarial example crafted with Fast Gradient Sign Method with L_2-norm and $\epsilon = 0.3$;

2. Virtual adversarial example (VAT) with $\epsilon = 0.3$;

3. Adversarial example crafted with Jacobian Saliency Method with feature adjustment $\theta = 0.1$;

4. Perturbed example drawn from the Uniform distribution centered in x ($x' \sim \mathcal{U}(x)$);

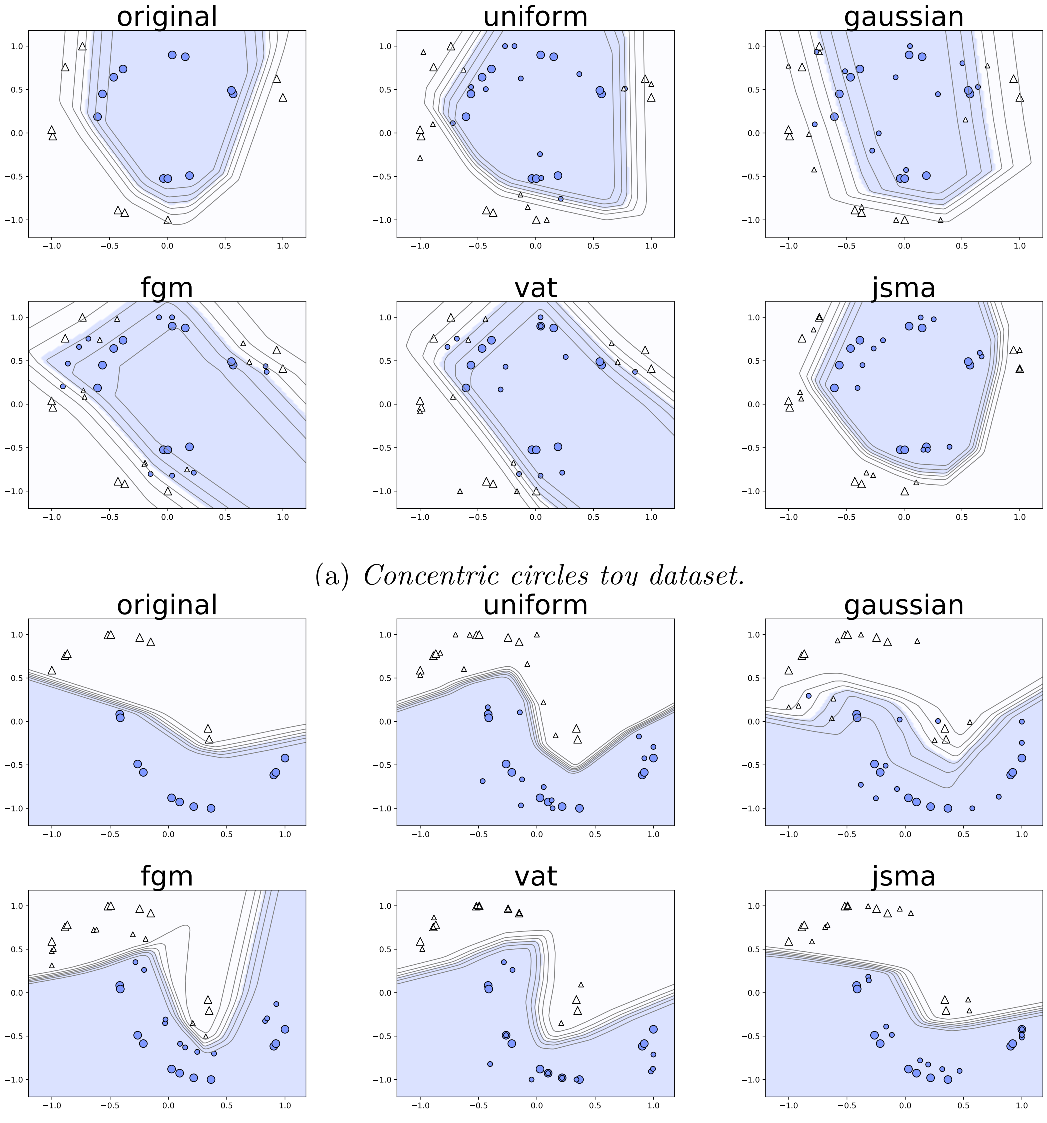
5. Perturbed example drawn from a Gaussian distribution centered in x, with standard deviation $\sigma = 0.3$ ($x' \sim \mathcal{N}(x, \sigma^2)$).

GDA helps smoothing the model confidence without affecting the accuracy on the true examples (sometimes even improving it), as shown in Figure B.2. Notice how the change in the value of the loss function is smoother for GDA than for the other defense methods. We also compare against augmentation with uniformly generated random noise, as was previously done in [136]. In this case as well, the GDA makes up for smoother variations.

These experiments confirm that, even though perturbing the true instances with random noise does not produce successful attacks [201], it is highly effective when deployed as a defense. Moreover, from a computational point of view, this new technique comes at practically no cost: the model does not require retraining, as opposed to adversarial training or defensive distillation, and drawing points from a Gaussian distribution is considerably cheaper than crafting adversarial examples.

B.4 Experiments

In this section, we discuss the experiments conducted for a closer look at the proposed defense methods, and contrast their performance against other defenses under multiple attacks. Following a brief description of the experimental protocol, we detail the results obtained for an extensive class of setups. The various settings and evaluation metrics are motivated to acquire better insights into the workings of adversarial misclassification and into the robustness of methods.



(a) *Concentric circles toy dataset.*

(b) *Half-moons toy dataset.*

Figure B.2 – *We compare state-of-the-art data augmentation techniques for hardening learning models, in this case a soft-max neural network with two dense hidden layers and RELU activation function. The decision boundary can be identified through the colors and the confidence level contours are marked with black lines. The original points and the additional ones (smaller) are drawn with the label and color corresponding to their class.*

B.4.1 Setup

Datasets Our experiments are performed on two standard machine learning datasets: MNIST [114] and CIFAR10 [100]. MNIST contains 70,000 samples of black and white 28×28 images, divided into 60,000 training samples and 10,000 test samples. Each pixel value is scaled between 0 and 1, and the digits are the 10 possible classes. CIFAR10 contains 60,000 images of size 32×32 with three color channels, their values also scaled between 0 and 1. The dataset is split into 50,000 training images and 10,000 test ones, all from 10 classes. We consider two types of network architectures: simple convolutional neural nets (CNN) [112] and convolutional neural nets with residuals (ResNet) [85]. The CNN is structured as follows: Conv2D(64, 8×8) – Conv2D(128, 6×6) – Conv2D(128, 5×5) – Softmax(10). The ResNet has the following layers with an identity short-cut between layer 2 and 6: Conv2D(64, 8×8) – Conv2D(128, 6×6) – Conv2D(64, 1×1) – Conv2D(64, 1×1) – Conv2D(128, 1×1) – MaxPooling(3×3) – Softmax(10). For both models, the activation function used is RELU, except when specified otherwise.

Methods We use the following attacks to craft adversarial examples, in view of comparing the respective capacities of defense models:

1. Fast Gradient Sign Method (FGSM) with distortion size ϵ ranging from 0.01 to 1, or with an iterative strategy determining the minimal perturbation necessary to change predictions;

2. FGSM applied after a preliminary step of adding Gaussian noise (within a range of $\alpha = 0.05$); the random noise applied in the first step is deducted from the budget of the FGSM attack; we call this heuristic Random + FGSM;

3. Jacobian Saliency Map Attack (JSMA) with the default parameters from the authors' code: $\gamma = 1$, $\theta = 0.1$;

4. DeepFool with a maximum of 100 iterations;

5. The L_2 attack from [38] (called C&W) with the default parameters from the authors' code and a confidence level of 2.3.

We compare the following defense methods:

1. Feature squeezing (FS), reducing the color depth to 1 bit for MNIST and 3 bits for CIFAR10, as the authors suggested in [210];

2. Label smoothing (LS), with the weight of the true label set to 0.9;

3. Adversarial training (AT) with examples crafted using FGSM with $\epsilon = 0.3$ for MNIST and $\epsilon = 0.05$ for CIFAR10;
4. Virtual adversarial training (VAT) with $\epsilon = 2.1$ as indicated in [136];
5. Gaussian Data Augmentation generating ten noisy samples for each original one, with standard deviation $\sigma = 0.3$ for MNIST and $\sigma = 0.05$ for CIFAR10; when BRELU ($t = 1$) is used as activation function, we call this method GDA + BRELU, otherwise GDA + RELU.

Notice that we do not compare directly against defensive distillation, but consider label smoothing instead, for the reasons given in Section B.2.

Measuring defense efficiency Most experimental studies evaluate the performance of defense strategies on the basis of their effect on classification accuracy. However, we believe it is also vital to measure the robustness of the obtained models to an attack [139], for instance by quantifying the average distortion introduced during adversarial generation. These metrics provide better insights into the local behavior of defenses. We compute three such metrics in our experiments: the empirical robustness (as measured by minimal perturbation), the distance to the training set and the loss sensitivity to pertubations. For a model F, the robustness is defined as:

$$\rho_F = \mathbb{E}_{(x,y)\sim\mathcal{D}} \frac{\Delta x}{||x||_2 + \epsilon}, \tag{B.2}$$

where Δx amounts to the L_2 perturbation that is required for an instance x in order for the model F to change its prediction under a certain attack. Here, ϵ is an extremely small constant allowing for the division to be defined. Intuitively, robust defenses require larger perturbations before the prediction changes. We estimate robustness by its empirical value computed on the available sample.

Training set distance offers a complementary viewpoint to robustness. In its attempt to quantify the dissimilarity between two sets, namely training data and adversarial images, this metric measures the average nearest-neighbour distance between samples from the two sets. Consequently, a larger value of this metric for adversarial images obtained by minimal perturbation supports the robustness of a defense strategy. Furthermore, this metric could also serve as a detection tool for adversarial attacks.

One of the main features of the proposed method is the smoothing effect on the learned model. We propose to quantify this smoothness by estimating the Lipschitz continuity constant ℓ of the model, which measures the largest variation of a function under a small change in its input: the lower the value, the smoother the function.

In practice, we are unable to compute this theoretical metric. We propose instead to estimate ℓ_F by local loss sensitivity analysis [101], using the gradients of the loss function with respect to the M input points in the test set:

$$\ell_F = \frac{1}{M}\sum_{i=1}^{M}\left\|\frac{\partial J(\theta, x_i, y_i)}{\partial x_i}\right\|_2 .$$

B.4.2 Comparison Between Architectures

In this experiment, we aim to show the impact of the type of deep network architecture against adversarial attacks. Namely, we compare CNN against ResNet, and RELU against BRELU activations respectively, under attacks crafted with FGSM. As a matter of fact, we would like to reject the hypothesis of vanishing units, for which the misclassification of the adversarial examples would be caused by the deactivation of specific units and not by the activation of the wrong ones. If that was the case, (i) getting residuals from the previous layers as in ResNet would result in higher robustness and (ii) using BRELU activation functions would decrease the performances of the model on adversarial examples. However, as Figure B.3 shows, ResNet does not sustain attacks better than CNN, suggesting that the accumulation of errors through the neural network is the main cause of misclassification. Notice that CNN + BRELU performs best for a distortion of up to $\epsilon = 0.3$, which has been suggested to be the highest value for which an FGSM attack might be undetectable.

Table B.1 – *Accuracy (%) on MNIST for FGSM attack transfer on different architectures ($\epsilon = 0.1$, best result in bold, second best in gray).*

Tested on Crafted on	CNN + RELU	CNN + BRELU	ResNet + RELU	ResNet + BRELU
CNN + RELU	73.88	90.16	**89.49**	**88.26**
CNN + BRELU	94.00	87.74	**90.73**	**90.02**
ResNet + RELU	**94.46**	**93.65**	61.88	75.94
ResNet + BRELU	**94.21**	**93.91**	78.70	58.51

We now analyze the same architectures for adversarial samples transferability: attacks crafted on each architecture with FGSM are applied to all the models (see Table B.1). When the source and the target are the same, this makes for a white-box attack; otherwise, it is equivalent to a black-box setting. As is expected, each architecture is most fooled by its own adversarial examples. Additionally, ResNet is overall more affected by adversarial examples transferred from different architectures, while the CNN retains an accuracy higher than 90% in all black-box attacks, even when the only change in the architecture is replacing RELU with BRELU. This makes the case once more for noise accumulation over the layer rendering models vulnerable to adversarial samples. Considering the results of the

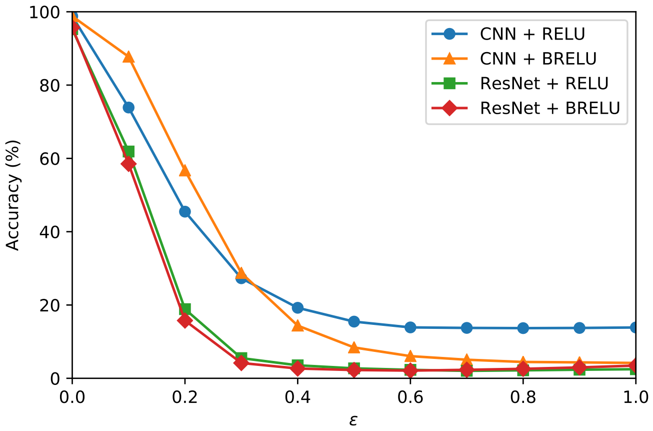
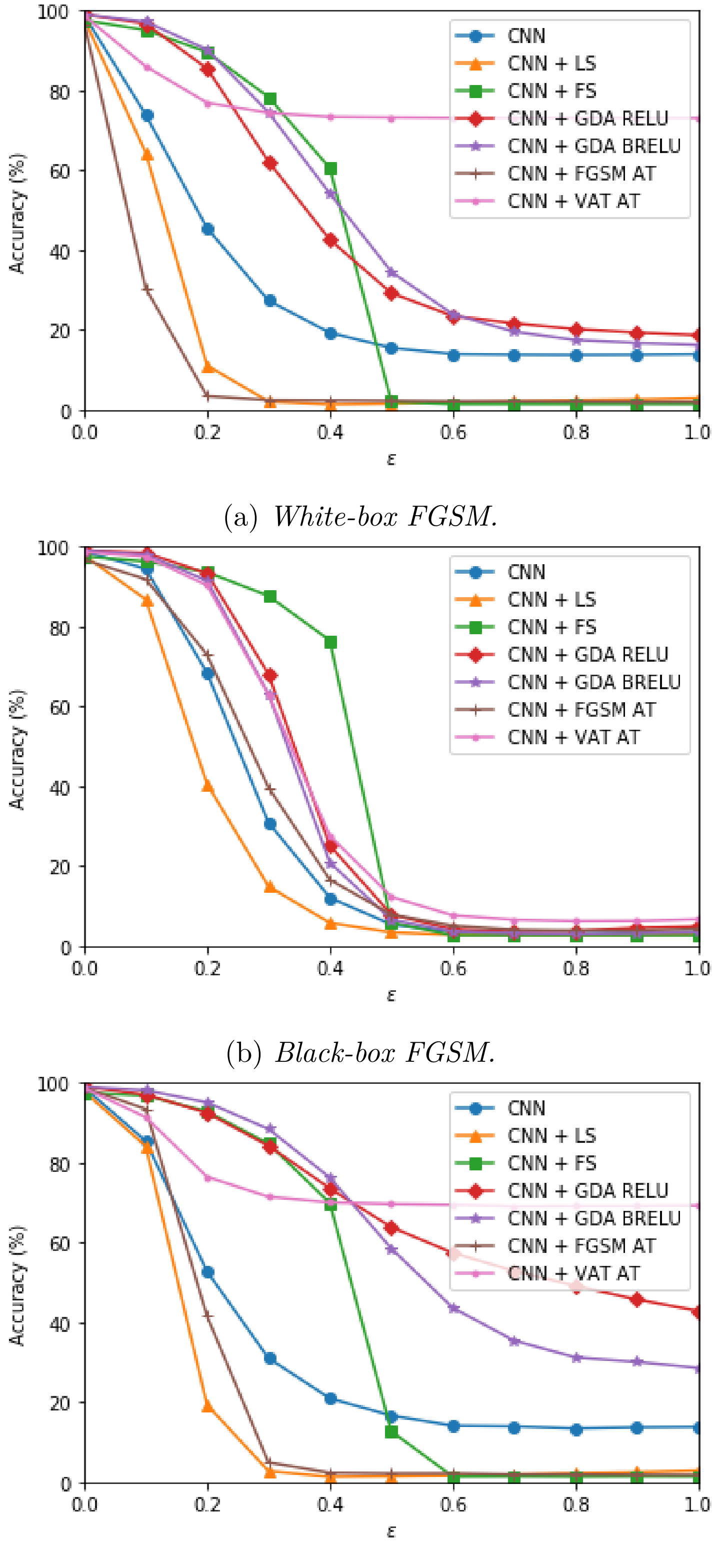


Figure B.3 – *Accuracy on FGSM white-box attack with respect to ϵ for different architectures on MNIST.*

ResNet model, we perform the remaining experiments only on the simple CNN architecture.

B.4.3 Impact of the Attack Distortion

We now study the impact of the attack distortion on the CNN model trained with different defenses. To this end, we use FGSM and Random + FGSM as attacks and plot the results in Figures B.4 and B.5. Note that for CIFAR10, we only show the results for $0 \leq \epsilon \leq 0.3$, as for bigger values, the accuracies are almost constant. The first observation is that label smoothing not only fails to strengthen the model in both setups, but it even worsens its robustness to adversarial attacks. Also, adversarial training seems ineffective even against white-box attacks, contrary to the observations in [185]. The result is not surprising, as the model is enforced on specific directions and probably loses its generalization capabilities. We explain this only apparent contradiction by pointing out the difference in the definitions of a white-box attack: for [185], white-box adversarial means examples crafted on the original model, without defenses; for us, they are crafted on the model trained through adversarial training. For MNIST, the three defenses that consistently perform the best are feature squeezing, virtual adversarial training and Gaussian data augmentation. However, if we focus on the results for $\epsilon < 0.3$ (which are the most significant, as big perturbations are easily detectable and result in rubbish examples), VAT is not as effective as the other two methods. On CIFAR10, small perturbations easily compromise the accuracy of the models even when defended by the methods the most efficient on MNIST, notably feature squeezing and virtual adversarial training. Moreover, using these last two defenses seems to degrade

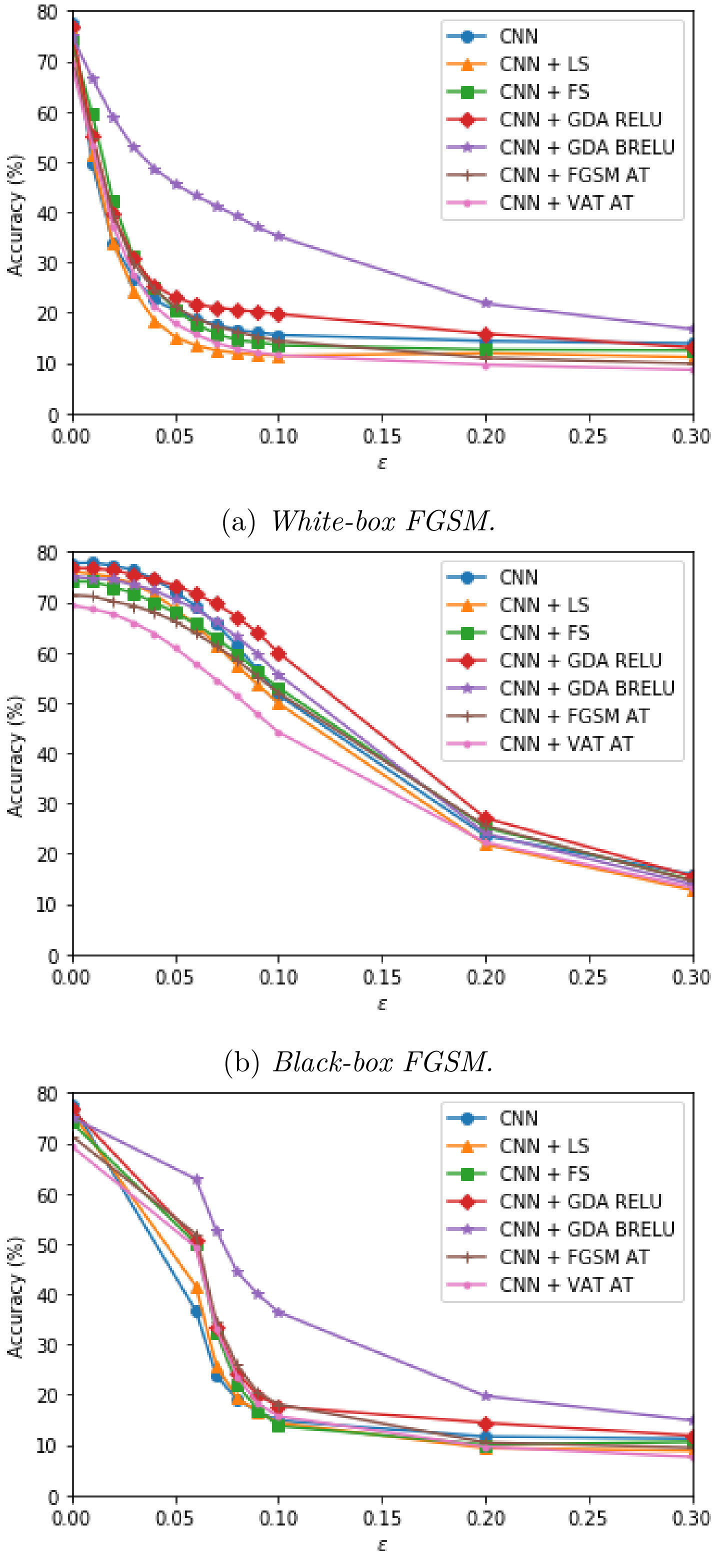


(a) *White-box FGSM.*

(b) *Black-box FGSM.*

(c) *White-box Random + FGSM.*

Figure B.4 – *Comparison of different defenses against white-box and black-box attacks on MNIST. For black-box attacks, the adversarial examples are crafted using the ResNet model, without any defense.*



(a) *White-box FGSM.*

(b) *Black-box FGSM.*

(c) *White-box Random + FGSM.*

Figure B.5 – *Comparison of different defenses against white-box and black-box attacks on CIFAR10. For black-box attacks, the adversarial examples are crafted using the ResNet model, without any defense. Note that the adversarial examples crafted with Random + FGSM and* $0 < \epsilon \leq 0.05$ *correspond to Gaussian noise for* $\alpha = 0.05$.

the performances on true examples and to fail in strengthening the model from black-box and white-box attacks alike.

In conclusion, our defenses outperform state-of-the-art strategies in terms of accuracy on these two datasets.

B.4.4 Defense Performance under Multiple Metrics

As discussed previously, the accuracy is not a sufficient measure for evaluating the performance of a model, especially in the case of adversarial samples: an attack might make for an incorrect prediction with respect to the original label in most cases, but if the adversarial examples cannot be mistaken as legitimate inputs, the attack is arguably not effective. For this reason, we propose to evaluate the robustness of defenses, as well as the shift of adversarial examples with respect to the clean data distribution. The adversarial examples are crafted incrementally using FGSM with small perturbations, and the algorithm stops when the prediction changes. The results in Table B.2 show that GDA with RELU as activation function performs best in terms of accuracy, being on a par with VAT. The robustness indicates the average amount of minimal perturbation necessary to achieve misclassification. This measure proves that both versions of the proposed defense yield a more robust model, potentially making the adversarial examples visually detectable. It is interesting to notice that feature squeezing and label smoothing actually decrease the robustness of the model. The last column in the table measures the average (Euclidean) distance of adversarial samples to the closest training point: a higher values indicates a larger shift between the two distributions. Here as well, GDA with both setups obtains much better results than the other defenses. Coupled with the robustness results, this metric confirms that one cannot have resistance to adversarial samples without an overall robustness of models, which calls for generic model reinforcement methods like the one proposed in this paper. Table B.3 presents the measure of the sensitivity of the loss function under small variations for all studied defense methods. Gaussian augmentation provides the smoothest model with the RELU activation. Notice that feature squeezing and label smoothing both induce higher gradients in the model, while the other defenses enforce a level of smoothness close to the one of the original unprotected model.

B.4.5 Transferability of Adversarial Samples

We now compare the performance of the proposed method against other defenses in a black-box setting, to account for the adversarial examples transferability phenomenon. To this end, all adversarial examples are crafted on the ResNet

Table B.2 – *White-box attack on adversarial examples from FGSM with minimal perturbation (best result in bold, second best in gray).*

Defense	Accuracy	Robustness	Training set distance
CNN	52.07	0.202	3.90
Feature squeezing	35.61	0.143	3.01
Label smoothing	37.06	0.152	3.18
FGSM adversarial training	56.44	0.226	4.33
VAT	**73.32**	0.308	6.64
GDA + RELU	**73.96**	**0.440**	**10.36**
GDA + BRELU	69.56	**0.471**	**9.31**

Table B.3 – *Local loss sensitivity analysis for defenses on MNIST (best result in bold, second best in gray).*

Model	Local sensitivity
CNN	**0.0609**
Feature squeeze	0.1215
Label smoothing	0.2289
FGSM adversarial training	0.0748
VAT	0.0741
GDA + RELU	**0.0244**
GDA + BRELU	0.0753

Table B.4 – *Accuracy (%) for black-box attacks on MNIST (best result in bold, second best in gray).*

Attack Defense	FGSM	Rand + FGSM	DeepFool	JSMA	C&W
CNN	94.46	40.70	92.95	97.95	93.10
Feature squeezing	96.31	**91.09**	96.68	97.48	96.75
Label smoothing	86.79	20.28	84.58	95.86	84.81
FGSM adversarial training	91.86	49.77	85.91	98.62	97.71
VAT	97.53	74.35	96.03	98.26	96.11
GDA + RELU	**98.47**	**80.25**	**97.84**	**98.96**	**97.87**
GDA + BRELU	**98.08**	75.50	**98.00**	**98.88**	**98.03**

Table B.5 – *Accuracy (%) for black-box attacks on CIFAR10 (best result in bold, second best in gray).*

Attack Defense	FGSM	Rand + FGSM	DeepFool	JSMA	C&W
CNN	51.55	57.22	**77.76**	**76.85**	**77.79**
Feature squeezing	52.98	61.83	73.82	73.28	73.75
Label smoothing	49.90	56.68	75.80	74.78	75.66
FGSM adversarial training	51.94	63.25	71.17	70.48	71.02
VAT	44.16	56.47	68.94	68.21	68.47
GDA + RELU	**59.96**	**71.93**	**76.80**	**76.45**	**76.61**
GDA + BRELU	**55.65**	**69.43**	74.97	74.73	74.78

model described previously, before being applied to the CNN model trained with each of the defense methods. In the case of the FGSM attack, ϵ is set to 0.1. Tables B.4 and B.5 present the classification accuracy obtained by the models. The first line is the baseline, that is the CNN with no defense. For our method, we consider GDA both with and without the use of BRELU. On MNIST, our methods outperform the other defenses in most cases. Namely, notice that they obtain the best performance under FGSM, VAT, DeepFool, JSMA and C&W attacks; this difference is significant for FGSM, DeepFool and C&W. Random + FGSM has been shown to be a stronger attack than one-step attacks; this is confirmed by the poor performance of all defenses, except for feature squeezing. An interesting fact is that using label smoothing degrades the performance of the model under the Random + FGSM attack when compared to the CNN with no defense. On CIFAR10, our methods consistently make the model more robust than the other defenses.

B.5 Conclusion

Despite their widespread adoption, deep learning models are victims of uninterpretable and counterintuitive behavior. While numerous hypotheses compete to provide an explanation for adversarial samples, their root cause still remains largely unknown. The quest of understanding this phenomenon has turned into an arms race of attack and defense strategies. Along with the drive for efficient and better attacks, there is a parallel hunt for effective defenses which can guard against them. Given this large ammo of strategies, practitioners are faced with a dire need for attack-agnostic defense schemes which can be easily employed at their end.

This work proposed two such strategies which, used separately or combined, improve the robustness of a deep model. We built on two intuitive hypotheses of error accumulation and smoothness assumption, and proposed to impose two constraints: first, on the architecture in the form of the bounded RELU activations, and second, in the form of training with Gaussian augmented data. We demonstrated the utility of this combined approach against the state-of-art attacks. Compared to adversarial training based on an attack, our defense has the major advantage of being computationally inexpensive; first, it only requires training one model, and second, Gaussian noise has much lower computational cost than crafting adversarial examples. The latter property allows us to explore a larger set of directions around each input than adversarial training. The overall effect is obtaining a smoother, more stable model, which is able to sustain a wide range of adversarial attacks. As we have shown in the experimental section, we achieve this without compromising on the original classification performance on clean input.

Lipschitz Continuity of Mahalanobis Distances and Bilinear Forms

C

This Annex contains the technical report

Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. Lipschitz continuity of mahalanobis distances and bilinear forms. 2016.

Many theoretical results in the machine learning domain stand only for functions that are Lipschitz continuous. Lipschitz continuity is a strong form of continuity that linearly bounds the variations of a function. In this paper, we derive tight Lipschitz constants for two families of metrics: Mahalanobis distances and bounded-space bilinear forms. To our knowledge, this is the first time the Mahalanobis distance is formally proved to be Lipschitz continuous and that such tight Lipschitz constants are derived.

C.1 Multi-variate Lipschitz continuity

A function is said Lipschitz continuous if it takes similar values on points that are close. More precisely, the slope of the function is bounded by a constant that is independent of the choice of points. This means that the variation of a function that is Lipschitz continuous within a certain interval is small. The Lipschitz continuity is a strong form of uniform continuity: for instance, a function that is Lipschitz continuous is also continuous, but the reverse is not necessarily true. Let's take the example of the square function: x^2 is continuous on $\mathbb{R}^d$ but it is not Lipschitz continuous (the slope of x^2 is not bounded).

We now consider the Lipschitz continuity for a function $f : \mathcal{X}^2 \subset \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.

Definition C.1 *(Multi-variate Lipschitz continuity) A function* $f : \mathcal{X}^2 \subset \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

is said k_p-lipschitz w.r.t. the norm $\|.\|_p$ if $\forall (x_1, x_2, x_1', x_2') \in \mathcal{X}^4$*:*

$$|f(x_1,x_2) - f(x_1',x_2')| \leq k_p \left\| \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} \right\|_p . \tag{C.1}$$

If f is differentiable on $\mathcal{X}^2 \subset \mathbb{R}^d \times \mathbb{R}^d$ and $\mathcal{X}^2$ is a convex space, the best constant k_p, the characteristic Lipschitz coefficient, can be estimated considering the fact that

$$\begin{aligned} k_p = & \sup_{x_1,x_2,x_1',x_2' \in \mathcal{X}} \left(\frac{|f(x_1,x_2) - f(x_1',x_2')|}{\left\| \binom{x_1}{x_2} - \binom{x_1'}{x_2'} \right\|_p} \right) = \\ = & \sup_{x_1,x_2 \in \mathcal{X}} \|\nabla f(x_1,x_2)\|_p . \end{aligned} \tag{C.2}$$

C.2 Derivation for particular functions

In this section, we analyze the Lipschitz continuity of two classic metric functions: the Mahalanobis distance and the bilinear form. These two functions are largely used in the field of Machine Learning, especially in Metric Learning.

C.2.1 Derivation for Mahalanobis-like Distances

We recall that the Mahalanobis distance of a pair $(x_1, x_2) \in \mathcal{X}^2$ can be written as $d_M(x_1,x_2) = \sqrt{(x_1 - x_2)^T M (x_1 - x_2)}$ where M is some Positive Semi-Definite matrix, whose coefficients can be optimized. By Def. C.1, the function $d_M : \mathcal{X}^2 \to \mathbb{R}$ is k_p-lipschitz w.r.t. the norm $\|.\|_p$ if $\forall x_1, x_2 \in \mathcal{X}$, $\|\nabla d_M(x_1,x_2)\|_p$ can be bounded by a constant k_p, where

$$\nabla d_M(x_1,x_2) = \begin{pmatrix} \frac{\partial d_M(x_1,x_2)}{\partial x_1} \\ \frac{\partial d_M(x_1,x_2)}{\partial x_2} \end{pmatrix}$$

and, for this particular case:

$$\begin{aligned} \frac{\partial d_M(x_1,x_2)}{\partial x_1} &= \frac{1}{2\sqrt{(x_1-x_2)^T M (x_1-x_2)}} \frac{\partial}{\partial x_1} \Big((x_1-x_2)^T M (x_1 - x_2) \Big) \\ &= \frac{1}{2\sqrt{(x_1-x_2)^T M (x_1-x_2)}} \frac{\partial}{\partial x_1} \left(x_1^T M x_1 - x_2^T M x_1 - x_1^T M x_2 + x_2^T M x_2 \right) \end{aligned} \tag{C.3}$$

$$= \frac{2Mx_1 - Mx_2 - Mx_2}{2\sqrt{(x_1-x_2)^T M (x_1-x_2)}} \tag{C.4}$$

$$= \frac{Mx_1 - Mx_2}{\sqrt{(x_1-x_2)^T M (x_1-x_2)}} = \frac{M(x_1 - x_2)}{\sqrt{(x_1-x_2)^T M (x_1-x_2)}}$$

and, in the same way:

$$\frac{\partial d_M(x_1,x_2)}{\partial x_2} = \frac{M(x_2-x_1)}{\sqrt{(x_1-x_2)^T M(x_1-x_2)}}.$$

In Eq. C.3 and C.4 we made use of the symmetry of the matrix M.

Lemma C.1 *The Mahalanobis distance $d_M(x_1,x_2)$ is k-lipschitz w.r.t. the norm $\|.\|_2$, with $k=\sqrt{2}\,\|L\|_2$, where $M=L^TL$, with L a lower triangular matrix.*

Proof.

$$\begin{aligned}
&\max_{x_1,x_2\in\mathcal{X}} \|\nabla d_M(x_1,x_2)\|_2 \\
&= \max_{x_1,x_2\in\mathcal{X}} \sqrt{\left\|\frac{\partial d_M(x_1,x_2)}{\partial x_1}\right\|_2^2 + \left\|\frac{\partial d_M(x_1,x_2)}{\partial x_2}\right\|_2^2} \\
&= \max_{x_1,x_2\in\mathcal{X}} \sqrt{\left\|\frac{M(x_1-x_2)}{\sqrt{(x_1-x_2)^T M(x_1-x_2)}}\right\|_2^2 + \left\|\frac{M(x_2-x_1)}{\sqrt{(x_1-x_2)^T M(x_1-x_2)}}\right\|_2^2} \\
&= \max_{x_1,x_2\in\mathcal{X}} \sqrt{2\left\|\frac{M(x_1-x_2)}{\sqrt{(x_1-x_2)^T M(x_1-x_2)}}\right\|_2^2} \\
&= \max_{x_1,x_2\in\mathcal{X}} \sqrt{2\left\|\frac{L^TL(x_1-x_2)}{\sqrt{(x_1-x_2)^T L^TL(x_1-x_2)}}\right\|_2^2} \qquad (C.5) \\
&= \max_{x_1,x_2\in\mathcal{X}} \sqrt{2\left\|\frac{L^T(L(x_1-x_2))}{\sqrt{(L(x_1-x_2))^T L(x_1-x_2)}}\right\|_2^2} \\
&= \max_{x_1,x_2\in\mathcal{X}} \sqrt{2\left\|L^T\frac{(L(x_1-x_2))}{\|L(x_1-x_2)\|_2}\right\|_2^2} \\
&\le \max_{x_1,x_2\in\mathcal{X}} \sqrt{2\,\|L^T\|_2\left\|\frac{(L(x_1-x_2))}{\|L(x_1-x_2)\|_2}\right\|_2^2} \qquad (C.6) \\
&\le \sqrt{2}\left\|L^T\right\|_2 = k. \qquad (C.7)
\end{aligned}$$

In Eq. C.5 we applied the Cholesky decomposition $M=L^TL$, the bound in C.6 is due to the Cauchy-Schwarz inequality and in Eq. C.7 $\left\|\frac{(L(x_1-x_2))}{\|L(x_1-x_2)\|_2}\right\|_2 = 1$ because it is a normalized vector.

□

C.2.2 Derivation for Bilinear Forms

We recall that the bilinear form of a pair (x_1, x_2) is computed as $d_M(x_1, x_2) = x_1^T M x_2$, where M is a generic matrix that can be optimized. Then:

$$\frac{\partial d_M(x_1, x_2)}{\partial x_1} = \frac{\partial}{\partial x_1}\left(x_1^T M x_2)\right) = M x_2$$

$$\frac{\partial d_M(x_1, x_2)}{\partial x_2} = \frac{\partial}{\partial x_2}\left(x_1^T M x_2)\right) = M^T x_1.$$

Lemma C.2 *The bilinear similarity $d_M(x_1, x_2) = x_1^T M x_2$ is k-lipschitz w.r.t. the norm $\|.\|_2$, with $k = \sqrt{2}\,\|M\|_2\,R$, when $\|x\|_2 \leq R\ \forall x \in \mathcal{X}$.*

Proof.

$$\begin{aligned}\max_{\forall x_1, x_2 \in U} \|\nabla d_M(x_1, x_2)\|_2 &= \max_{\forall x_1, x_2 \in U} \sqrt{\left\|\frac{\partial d_M(x_1, x_2)}{\partial x_1}\right\|_2^2 + \left\|\frac{\partial d_M(x_1, x_2)}{\partial x_2}\right\|_2^2} \\ &= \max_{\forall x_1, x_2 \in U} \sqrt{\|M x_2\|_2^2 + \|M^T x_1\|_2^2} \\ &\leq \sqrt{2}\,\|M\|_2\,R = k. \qquad \text{(C.8)}\end{aligned}$$

□

C.3 Conclusion

In this paper, we recalled a method for proving the Lipschitz continuity and for finding a tight Lipschitz constant of multivariate differentiable functions. Using this approach, we computed tight Lipschitz constants for two families of metrics that are heavily used, especially in metric learning. We have shown that the Mahalanobis distance is Lipschitz continuous and has a constant of $\sqrt{2}\,\|L\|_2$ (where L is the square root of the correlation matrix). We have also shown that the bilinear form xMy is Lipschitz continuous with a constant $\sqrt{2}\,\|M\|_2\,R$ (when the space is bounded by R).

Many theoretical results in the machine learning domain rely on Lipschitz continuity and depend on the Lipschitz constants. For example, the generalization bounds obtained in the context of the uniform stability (see [17]) can be derived by constraining the studied functions to be Lipschitz continuous and the tightness of those bounds depends on the value of the Lipschitz constant. The derivations from this paper have been originally developed to derive theoretical bounds for [219]. We believe these results can also be used to derive tighter theoretical bounds in other domains of machine learning.

Appendix of Chapter 4 D

D.1 Bound of product space curvature

We first recall our notations. For any $k \in [K]$, we let $\mathcal{M}^k = \{\alpha_k \in \mathbb{R}^n : \left\|\alpha_k\right\|_1 \leq \beta\}$ and denote by $\mathcal{M} = \mathcal{M}^1 \times \cdots \times \mathcal{M}^K$ our feasible domain in (D.1). For convenience, we will use $\alpha = [\alpha_1, \ldots, \alpha_K] \in (\mathbb{R}^n)^K$ to denote the concatenation of the local classifiers $\{\alpha_k\}_{k=1}^K$ and refer to the objective function (D.1) as $f(\alpha)$. We also denote by $v_{[k]} \in \mathcal{M}$ the zero-padding of any vector $v_k \in \mathcal{M}^k$.

We recall our joint optimization problem over the classifiers $\alpha_1, \ldots, \alpha_K$:

$$\begin{aligned} \min_{\|\alpha_1\|_1 \leq \beta, \ldots, \|\alpha_K\|_1 \leq \beta} f(\alpha_1, \ldots, \alpha_K) = & \sum_{k=1}^K D_k c_k \log\left(\frac{1}{m_k}\sum_{i=1}^{m_k} \exp\left(-(A_k\alpha_k)_i\right)\right) \\ & + \frac{\mu}{2}\sum_{k=1}^K\sum_{l=1}^{k-1} W_{kl}\|\alpha_k - \alpha_l\|^2. \end{aligned} \tag{D.1}$$

Lemma D.1 *For Problem (D.1), we have $C_f^{\otimes} \leq 4\beta^2 \sum_{k=1}^K (D_k c_k \left\|A_k\right\|_1^2 + \mu D_k)$.*

Proof. For the following proof, we utilize two key concepts of functional analysis: the Lipschitz continuity and the diameter of a compact space. A function $f : \mathcal{X} \to \mathbb{R}$ is L-lipschitz w.r.t. the norm $\left\|.\right\|_1$ if $\forall (x, x') \in \mathcal{X}^2$:

$$|f(x) - f(x')| \leq L\left\|x - x'\right\|_1. \tag{D.2}$$

The diameter of a compact normed vector space $(\mathcal{M}, \|.\|)$ is defined as:

$$\operatorname{diam}_{\|.\|}(\mathcal{M}) = \sup_{x, x' \in \mathcal{M}} \left\|x - x'\right\|. \tag{D.3}$$

We recall the formulation of the partial gradient

$$\nabla_k f(\alpha) = -D_k c_k w_k(\alpha)^T A_k + \mu\left(D_k \alpha_{[k]} - \sum_l W_{kl}\alpha_{[l]}\right),$$

where we denote $w_k(\alpha) = \frac{\exp(-A_k\alpha_{[k]})}{\sum_{i=1}^{m_k}\exp(-A_k\alpha_{[k]})_i}$.

We bound the Lipschitz constant of $w_k(\alpha)$ by bounding its first derivative:

$$\begin{aligned} \left\|\nabla_k(w_k(\alpha))\right\|_1 &= \left\|(-w_k(\alpha) + w_k(\alpha)^2)^T A_k\right\|_1 \\ &\leq \left\|A_k\right\|_1. \end{aligned} \tag{D.4}$$

Eq. (D.4) is due to the fact that $\left\|w_k\right\|_1 \leq 1$ and $w_k \geq 0$. It is then easy to see that considering any two vectors $\alpha, \alpha' \in \mathcal{M}$ differing only in their k-th block ($\alpha_{[l]} = \alpha'_{[l]}\ \forall l \neq k$), the Lipschitz constant of the partial gradient $\nabla_k f$ in (D.4) is bounded by $L_k = D_k c_k \|A_k\|_1^2 + \mu D_k$.

We can easily bound the diameter of the subspace $\mathcal{M}^k = \{\alpha_k \in \mathbb{R}^n : \left\|\alpha_k\right\|_1 \leq \beta\}$ as follows:

$$\mathrm{diam}_{\|.\|_1}(\mathcal{M}^k) = \max_{\alpha_k, \alpha'_k \in \mathcal{M}^k} \left\|\alpha_k - \alpha'_k\right\|_1 = 2\beta.$$

Finally, we obtain Lemma D.1 by combining the above results:

$$C_f^{\otimes} = \sum_{k=1}^{K} C_f^k \leq \sum_{k=1}^{K} L_k \,\mathrm{diam}^2_{\|.\|_1}(\mathcal{M}^k) = 4\beta^2 \textstyle\sum_{k=1}^{K}(D_k c_k \left\|A_k\right\|_1^2 + \mu D_k). \tag{D.5}$$

□

Appendix of Chapter 5

E

E.1 Hilbert space $\mathcal{H}$

We recall the definition of Hilbert Space.

Definition E.1 *(Hilbert Space) A real vector space $\mathcal{V}$ over $\mathbb{R}$ is a Hilbert Space if:*

1. *$\mathcal{V}$ is a real inner product space;*
2. *$\mathcal{V}$ is a complete metric space with respect to the distance function induced by its inner product.*

Theorem E.1 *The space $\mathcal{H}$ resulting by a transformation $\mu_{\mathcal{L}}(x) = [\mu(x, l_1), ..., \mu(x, l_L)]$, with $\mu : \mathcal{X}^2 \to \mathbb{R}$ of an Hilbert space $\mathcal{X}$ is also an Hilbert Space if $\mathcal{L} \neq \mathbf{0}$.*

Proof. If $\mathcal{L} \neq \mathbf{0}$, $<\mu_{\mathcal{L}}(), \mu_{\mathcal{L}}()>= \mu_{\mathcal{L}}()\mu_{\mathcal{L}}()^T$ is an inner product, as:

1. $<\mu_{\mathcal{L}}(), \mu_{\mathcal{L}}()>$ is linear: $\forall a, b \in \mathbb{R}$ and $\forall x_1, x_2, x_3 \in \mathcal{X}$

$$\begin{aligned}
&< a\mu_{\mathcal{L}}(x_1) + b\mu_{\mathcal{L}}(x_2), \mu_{\mathcal{L}}(x_3) > \\
&\quad = \Big(a\mu_{\mathcal{L}}(x_1) + b\mu_{\mathcal{L}}(x_2)\Big)\mu_{\mathcal{L}}(x_3)^T \\
&\quad = a\mu_{\mathcal{L}}(x_1)\mu_{\mathcal{L}}(x_3)^T + b\mu_{\mathcal{L}}(x_2)\mu_{\mathcal{L}}(x_3)^T \\
&\quad = a < \mu_{\mathcal{L}}(x_1), \mu_{\mathcal{L}}(x_3) > +b < \mu_{\mathcal{L}}(x_2)\mu_{\mathcal{L}}(x_3) >;
\end{aligned}$$

2. $<\mu_{\mathcal{L}}(), \mu_{\mathcal{L}}()>$ is symmetric: $\forall x_1, x_2 \in \mathcal{X}$

$$< \mu_{\mathcal{L}}(x_1), \mu_{\mathcal{L}}(x_2) >=< \mu_{\mathcal{L}}(x_2), \mu_{\mathcal{L}}(x_1) >;$$

3. $<\mu_{\mathcal{L}}(), \mu_{\mathcal{L}}()>$ is always non-negative and null only for $\boldsymbol{x} = \mathbf{0}$: $\forall x \in \mathcal{X}$

$$< \mu_{\mathcal{L}}(x), \mu_{\mathcal{L}}(x) >= \sum_{p=1}^{L} \mu(x, p)^2 \geq 0$$

and $< \mu_{\mathcal{L}}(x), \mu_{\mathcal{L}}(x) >= 0$ iff $\boldsymbol{x} = \mathbf{0}$ as $\mathcal{L} \neq \mathbf{0}$.

□

In particular, the space generated by $\mu(x_1, x_2) = x_1^T x_2$ or $\mu(x_1, x_2) = \exp(-\frac{\|x_1 - x_2\|_2^2}{\sigma})$ is an Hilbert Space.

E.2 Lagrangian dual problem

The **L^3-SVMs** optimization problem takes the following form:

$$\arg\min_{\theta,b,\xi} \frac{1}{2} \|\theta\|_{\mathcal{F}}^2 + \frac{c}{m} \sum_{i=1}^{m} \xi_i$$

$$s.t.\; y_i \left(\theta_{k_i.} \mu_{\mathcal{L}}(x_i)^T + b\right) \geq 1 - \xi_i \;\; \forall i = 1..m$$

$$\xi_i \geq 0 \;\; \forall i = 1..m$$

with $\mu_{\mathcal{L}}(.) = [\mu(., l_1), ..., \mu(., l_L)]$ the projection from the input space $\mathcal{X}$ to the landmark space $\mathcal{H}$.

The Lagrangian dual problem of the previous formulation is obtained by maximizing the corresponding Lagrangian w.r.t. its Lagrangian multipliers. The derived problem is a Quadratic Programming problem that can be solved by common optimization techniques and that allows one to make use of the kernel trick. The Lagrangian takes the following form:

$$\mathcal{L}(\theta, b, \xi, \alpha, r) = \frac{1}{2} \|\theta\|_{\mathcal{F}}^2 + \frac{c}{m} \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} r_i \xi_i - \sum_{i=1}^{m} \alpha_i \left(y_i \left(\theta_{k_i.} \mu_{\mathcal{L}}(x_i)^T + b\right) + \xi_i - 1\right)$$

where $\alpha \in \mathbb{R}^m$ and $r \in \mathbb{R}^m$ are the positive Lagrangian multipliers. Let's consider the fact that:

$$\max_{\alpha,r} \min_{\theta,b,\xi} \mathcal{L}(\theta, b, \xi, \alpha, r) \leq \min_{\theta,b,\xi} \max_{\alpha,r} \mathcal{L}(\theta, b, \xi, \alpha, r)$$

where the left term corresponds to the optimal value of the dual problem and the right one to the primal's one. The dual and the primal problems have the same value at optimality if the Karush-Kuhn-Tucker (KKT) conditions are not violated (see [32]).

By setting the gradient of $\mathcal{L}$ w.r.t. θ, b and ξ to 0, we find the saddle point corresponding to the function minimum:

$$\nabla_{\theta_{kp}} \mathcal{L}(\theta, b, \xi, \alpha, r) = \theta_{kp} - \sum_{i=1|k_i=k}^{m} \alpha_i y_i \mu(x_i, l_p)$$

$$\nabla_b \mathcal{L}(\theta, b, \xi, \alpha, r) = -\sum_{i=1}^{m} \alpha_i y_i$$

$$\nabla_{\xi_i} \mathcal{L}(\theta, b, \xi, \alpha, r) = \frac{c}{m} - \alpha_i - r_i$$

which give

$$\theta_{kp} = \sum_{i=1|k_i=k}^{m} \alpha_i y_i \mu(x_i, l_p) \tag{E.1}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{E.2}$$

$$\alpha_i = \frac{c}{m} - r_i \tag{E.3}$$

We can now write the QP dual problem by replacing θ by its expression (E.1) and simplifying following (E.2) and (E.3):

$$\max_{\alpha} \; -\frac{1}{2} \sum_{i=1|k_i=k}^{m} \sum_{j=1|k_j=k}^{m} \alpha_i \alpha_j y_i y_j \mu_{\mathcal{L}}(x_i) \mu_{\mathcal{L}}(x_j)^T + \sum_{i=1}^{m} \alpha_i$$

$$s.t. \; 0 \leq \alpha_i \leq \frac{c}{m} \; \forall i = 1..m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \; \forall i = 1..m$$

which is concave w.r.t. α.

We need the following two additional constraints in order to respect the KKT conditions which guarantee that the optimal value found by solving the dual problem corresponds to the optimal value of the primal:

$$\alpha_i \left(y_i \left(\theta_{k_i.} \mu_{\mathcal{L}}(x_i)^T + b \right) - 1 + \xi_i \right) = 0 \; \forall i = 1..m$$

$$r_i \xi_i = 0 \; \forall i = 1..m$$

Once the Lagrangian dual problem solved, the characteristic vector θ and offset b of the optimal margin hyperplane can be retrieved by means of the support vectors, i.e. the instances whose corresponding α_i are strictly greater than 0:

$$\theta_{kp} = \sum_{a=1|k_a=k}^{m} \alpha_a y_a \mu(x_a, l_p)$$

$$b = y_a - \theta_{k_a.} \mu_{\mathcal{L}}(x_a)$$

and the new instances can be classified :

$$y(x) = sign\left(\theta_{k_i.} \mu_{\mathcal{L}}(x_i)^T + b \right).$$

E.3 Graphical representation of variable dependencies

Figures E.1 through E.3 graphically illustrates the variables involved in the different optimization problems that are solved by the local SVM approaches and **L^3-SVMs**. In these graphs, a node represents a variable (or a set of) and a link show a direct dependency between the variables, i.e., one variable is directly involved in the computation or the estimation of the other.

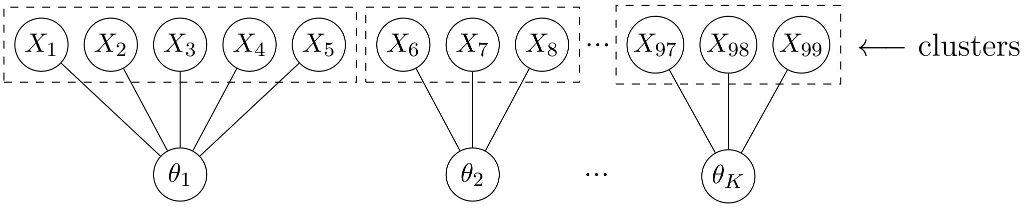


Figure E.1 – *Variable dependencies when learning one SVM per cluster (baseline used in Clustered SVM [80]).*

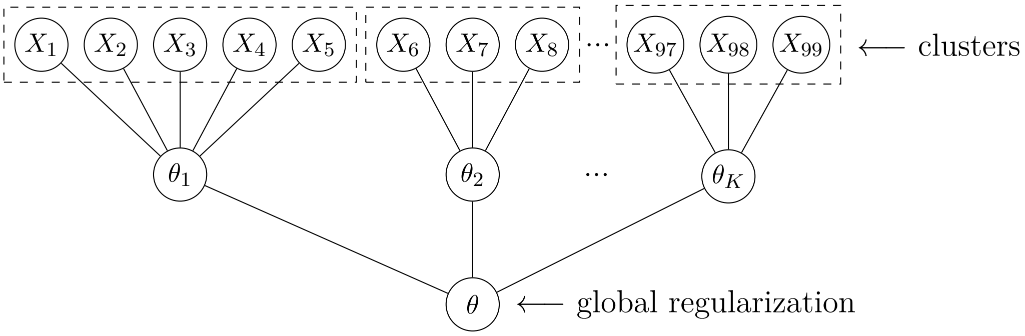


Figure E.2 – *Variable dependencies for Clustered SVM [80], where a common global regularization is used.*

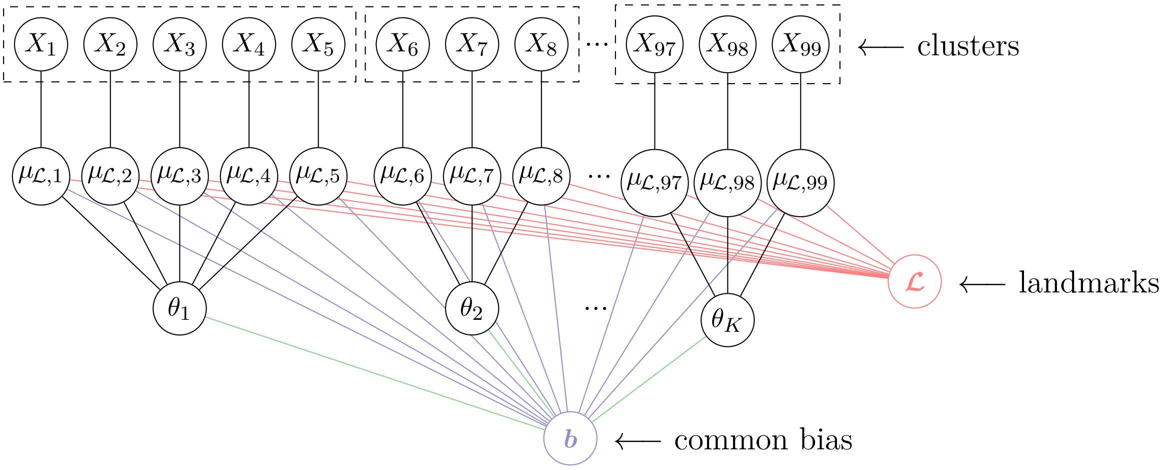


Figure E.3 – *Variable dependencies for our model, **L^3-SVMs**, where one SVM is learned per cluster but the local models interact through a common bias and $\mathcal{L}$, the set of landmarks.*

French Translations F

Table des Matières

F.1 Introduction

L'apprentissage machine est un domaine de l'intelligence artificielle qui englobe les algorithmes permettant à un ordinateur d'effectuer une tâche sans être programmé manuellement. En général, la structure et les caractéristiques de l'ensemble d'exemples recueillis sont analysées afin d'extrapoler de nouvelles informations, d'estimer la probabilité de certains événements et de prendre des décisions en conséquence. Une tâche typique de l'apprentissage machine est ce qu'on appelle la classification : à partir d'un ensemble d'observations (par exemple des corpus de textes) associées à un résultat cible (par exemple l'auteur), il est possible d'apprendre à prédire les résultats pour de nouvelles observations (par exemple, l'auteur de documents anonymes). En pratique, des modèles mathématiques sont déduits pour représenter les relations entre les inputs et les valeurs attendues ou, en général, pour estimer le mécanisme inconnu qui génère les données.

Habituellement, les problèmes sont abordés en les formulant mathématiquement : les données sont représentées sous forme de points dans un espace de caractéristiques, où chaque caractéristique (ou dimension) mesure un attribut particulier des instances ; l'objectif souhaité de l'apprentissage est décrit par une fonction qui guide la sélection du modèle. Les données d'entraînement ne sont pas simplement mémorisées. Comme nous avons accès à un échantillon limité de la distribution sous-jacente des exemples, la mémorisation de l'échantillon de formation entraînerait un mauvais rendement face à de nouvelles situations. Au lieu de cela, des informations utiles sont extraites de l'ensemble de données afin de bien généraliser sur les données futures, c'est-à-dire avoir de bonnes performances sur tout échantillon de la distribution inconnue.

Des exemples d'applications des algorithmes d'apprentissage machine peuvent être trouvés dans divers domaines, de la vision par ordinateur (pour le tracking d'objets, la reconnaissance faciale, etc.) au traitement du signal (par exemple la reconnaissance vocale) et ainsi de suite, menant à des systèmes complexes intégrant différents composants d'apprentissage machine, comme les assistants personnels virtuels et les véhicules autonomes. Les algorithmes d'apprentissage machine sont particulièrement utiles et préférables à leurs équivalents analytiques dans les situations suivantes :

- Le problème est difficile à décrire exactement, donc aucune méthodologie analytique n'est disponible pour le résoudre. Un exemple est la tâche de détection d'objets dans les images. Comme les objets et les concepts qui les définissent ne sont pas facilement dépeints (dans le sens où les descripteurs possibles ne peuvent pas être traduits du langage naturel à des formes logiques exhaustives) et que les conditions visuelles varient les caractéristiques des objets, aucune méthode exacte n'existe pour ces problèmes même si la détection des objets est une compétence innée des êtres humains.
- C'est plus rapide et moins coûteux d'apprendre une solution raffinée que de la coder en dur, car cette dernière nécessite plus d'expertise humaine et plus de travail. En computer graphics, par exemple, il peut être plus pratique d'apprendre à générer des textures procédurales diverses que de définir manuellement l'ensemble de la variété.
- Le problème peut varier avec le temps. Prenons le cas des attaques évasives contre les détecteurs de spams. Les attaquants s'adaptent constamment aux changements du système pour continuer d'échapper au filtre. Les solutions d'apprentissage machine offrent des moyens dynamiques pour prendre en compte de cette dérive conceptuelle.

Il existe plusieurs paradigmes pour résoudre ces tâches complexes. Cette thèse porte sur l'amélioration des approches d'apprentissage locales, une famille de techniques qui infère des modèles en capturant les caractéristiques locales de l'espace dans lequel les observations sont représentées. L'hypothèse fondatrice de ces techniques est que le modèle appris doit se comporter de manière cohérente sur des exemples qui sont proches, ce qui implique que ses résultats doivent aussi changer de façon continue dans l'espace. La localité peut être définie sur la base de critères spatiaux (par exemple, la proximité en fonction d'une métrique choisie) ou d'autres relations fournies, telles que l'association à la même catégorie d'exemples ou à un attribut commun.

On sait que les approches locales d'apprentissage sont efficaces pour capturer des distributions complexes de données, évitant de recourir à la sélection d'un modèle spécifique pour la tâche. Cependant, les techniques de pointe souffrent de trois inconvénients majeurs : elles mémorisent facilement l'ensemble d'entraînement, ce qui se traduit par des performances médiocres sur de nouvelles données ; leurs prédictions manquent de continuité dans des endroits particuliers de l'espace ; elles évoluent mal avec la taille des ensembles de données. Les contributions de cette thèse examinent les pièges susmentionnés dans deux directions : nous proposons d'introduire des informations secondaires dans la formulation du problème pour renforcer la continuité de la prédiction et atténuer le phénomène de mémorisation ; nous fournissons une nouvelle représentation de l'ensemble de données qui tient compte de ses spécificités locales et améliore son passage à l'échelle.

Contexte de la thèse Cette thèse a été réalisée au sein de l'équipe Data Intelligence du Laboratoire Hubert Curien UMR CNRS 5516, affilié à l'Université Jean Monnet de Saint-Etienne et l'Université de Lyon, France. Ses travaux ont été fondés par l'Agence Nationale de la Recherche à travers les projets de l'ANR SOLSTICE. (ANR-13-BS02-01), qui vise à concevoir de nouveaux modèles et outils pour faire face aux tâches de vision par ordinateur, et LIVES[1]. (ANR-15-CE23-0026-03), qui vise à développer des cadres d'apprentissage machine bien fondés pour l'apprentissage avec des données observées dans des vues multiples.

Aperçu du manuscrit Ce manuscrit est organisé en trois parties principales et plusieurs annexes. Par souci de cohérence, le corps principal de la thèse couvre les contributions liées à l'apprentissage local. Les autres contributions relevantes sont résumées ci-dessous et sont présentées en détail dans les annexes.

La partie I donne un aperçu du contexte scientifique et de l'état de l'art relatifs aux travaux présentés :

- Le Chapitre 1 présente le domaine de l'apprentissage statistique, avec ses hypothèses et ses principes, ainsi que ses défis et ses pratiques. Plus précisément, il détaille le déroulement du processus d'apprentissage des modèles statistiques capables de décrire les observations disponibles et d'avoir de bonnes performances sur de nouvelles données, ainsi que les cadres génériques pour étudier les capacités de généralisation d'un modèle appris, au-delà de sa performance empirique ;

[1] https://lives.lif.univ-mrs.fr/

- Le Chapitre 2 présente les principales solutions permettant d'améliorer les modèles linéaires, qui tirent parti du passage á l'échelle des modèles linéaires tout en étant capables de capturer des distributions de données complexes. Le chapitre se concentre particulièrement sur les approches locales de l'apprentissage, auxquelles appartiennent les contributions de cette thèse, et met en lumière leurs intuitions, leurs avantages et leurs pièges.

Part II collecte les contributions sur l'apprentissage local basé sur le partitionnement des données :

- Le Chapitre 3 décrit une nouvelle méthode d'apprentissage métrique pour surmonter les problèmes bien connus de l'apprentissage métrique local, à savoir leur tendance à la suradaptation et leur applicabilité limitée. L'approche apprend pour chaque paire de régions de l'espace d'entrée des combinaisons convexes de modèles locaux précédemment appris. De plus, la régularité de la prévision est renforcée par une régularisation spatiale qui encourage les modèles des régions voisines à être similaires. L'objectif est d'obtenir une similitude ou une distance pour comparer n'importe quelle paire de points qui soit adaptée aux régions auxquelles les points appartiennent. Une évaluation théorique de l'approche proposée est réalisée dans le cadre de la robustesse algorithmique pour la dérivation de bornes de généralisation. De plus, la méthode est comparée empiriquement aux techniques de l'état de l'art en termes de précision de régression sur deux tâches de régression. Ce travail a conduit aux publications **CVPR16** [219] et **CAp16** [216].
- Le Chapitre 4 propose une nouvelle technique décentralisée pour l'apprentissage collaboratif de modèles personnalisés sur un graphe d'utilisateurs qui est à la fois efficace et présentant un coût de communication limité. Le problème d'apprentissage prend la forme d'un l_1-Adaboost régularisé capable de construire des modèles non linéaires expressifs qui prennent en compte les singularités des données de chaque utilisateur mais qui ont aussi tendance à suivre les décisions des utilisateurs voisins dans le graphe. Le problème d'optimisation associé minimise conjointement l'erreur empirique globale tout en assurant la fluidité des modèles des utilisateurs par rapport au graphe de similarité. Nous proposons un algorithme décentralisé basé sur l'algorithme de Frank-Wolfe, exploitant la parcimonie intrinsèque des mises à jour pour obtenir une procédure d'apprentissage collaboratif efficace à faible coût de communication. De plus, une procédure de découverte de graphe est proposée pour estimer le graphe de similarité entre les utilisateurs s'il est inconnu. L'algorithme est analysé en fonction de son taux de convergence et de ses complexités de communication et de mémoire. Enfin, un ensemble

d'expériences est réalisé sur un jeu de données synthétique pour démontrer l'efficacité de la méthode proposée tant pour l'apprentissage des modèles personnalisés que pour l'estimation du graphe de communication. Ces contributions ont été présentées à **CAp18** [215] et à **MLPCD18**.

La partie III rassemble les contributions basées sur les similarités des points de repère :

- Le Chapitre 5 aborde les pièges des méthodes à noyau en introduisant une méthode locale basée sur les Support Vector Machines qui découpe l'espace d'entrée, projette les données sur les points de repère, et apprend conjointement une combinaison linéaire de modèles locaux. L'approche définit un mappage explicite à un espace latent où le problème est linéairement séparable. Ce faisant, l'approche passe mieux à l'échelle par rapport aux SVMs standards basés sur les fonctions à noyau. En utilisant le cadre de la stabilité uniforme, nous montrons que notre formulation du SVMs s'accompagne de garanties de généralisation sur le risque réel. Les expériences basées sur la configuration la plus simple de notre modèle (i.e. sélection aléatoire des points de repère, projections linéaires, noyau linéaire) montrent ses performances compétitives par rapport à l'état de l'art et ouvre la porte à de nouvelles lignes de recherche passionnantes. Ce travail a été présenté à **CAp17** [220].

- Le Chapitre 6 étend la méthode proposée dans le Chapitre 5 à une classification multi-vues. L'objectif est d'exploiter les informations complémentaires des différentes vues et de les mettre à l'échelle linéairement en fonction de la taille de l'ensemble des données. Les estimations de similarité par rapport à un petit ensemble de points de repère choisis au hasard sont effectués une vue à la fois, avant d'apprendre un SVM linéaire dans cet espace latent qui réunit toutes les vues. Selon le cadre de stabilité uniforme, l'algorithme proposé est robuste à de légères modifications de l'ensemble d'apprentissage, ce qui conduit à une borne de généralisation qui dépend du nombre de vues et de repères. Ce chapitre montre comment l'approche décrite peut être facilement adaptée à un scénario de vues manquantes en ne reconstruisant que les similarités avec les points de repère. Les résultats empiriques, dans les deux cadres des vues complètes ou manquantes, mettent en évidence les performances supérieures de la méthode proposée par rapport aux techniques de l'état de l'art, en termes de précision et de temps d'exécution. Ce travail a été publié sur **ECML18** [221].

Dans les annexes IV, nous présentons deux contributions élaborées au cours de la thèse mais sans rapport avec l'axe de l'apprentissage local :

- L'Annexe A présente un cadre générique pour apprendre à partir de données faiblement étiquetées. Ce domaine couvre différents contextes tels que l'apprentissage semi-supervisé, l'apprentissage avec des proportions d'étiquettes, l'apprentissage multi-instance, l'apprentissage tolérant au bruit, et ainsi de suite. L'annexe présente une nouvelle formulation du risque qui se résume à une généralisation du risque empirique standard basé sur des fonctions de perte dites surrogate. Le nouveau risque permet d'exprimer la fiabilité des étiquettes dans la fonction objectif et peut être utilisé pour dériver différents types d'algorithmes d'apprentissage, en fonction des connaissances sur les étiquettes. Ces travaux ont été publiés sur **NeurIPS16** [217].

- L'Annexe B introduit une nouvelle méthode de défense contre les attaques évasives adversarielles sur les réseaux neuronaux profonds, basée sur des observations pratiques. La défense proposée est facile à intégrer dans les modèles et plus performante que les défenses de l'état de l'art : elle est conçue pour renforcer la structure d'un DNN, rendant sa prédiction plus stable et moins susceptible d'être trompée par des échantillons adverses. L'annexe fait état d'une étude expérimentale approfondie prouvant l'efficacité de la méthode contre les attaques multiples, en comparaison avec de nombreuses défenses, à la fois en white-box et en black-box. Les travaux ont abouti à une publication à **AISEC17** [222] et à la publication de la bibliothèque open-source ART [145] pour étudier la robustesse adversarielle des DNNs.

Enfin, nous rapportons les notions et les preuves nécessaires à l'exhaustivité de trois chapitres :

- L'Appendice C présente les dérivations des constantes de Lipschitz pour deux familles de métriques que nous utilisons au Chapitre 3 : des distances de Mahalanobis et des formes bilinéaires. L'annexe présente la pré-publication [218].

- L'Appendice D présente la dérivation de la limite supérieure de la courbure de l'espace-produit nécessaire au Chapitre 4.

- L'Appendice E présente la dérivation du double problème lagrangien et quelques résultats additionnels relatifs au Chapitre 5.

F.2 Résumé des chapitres

F.2.1 Chapitre 1

Nous commençons ce manuscrit en donnant un aperçu du contexte scientifique de cette thèse : le domaine de l'apprentissage statistique englobe toutes les techniques visant à déduire une fonction prédictive des données à l'aide d'outils statistiques. Ce chapitre passe en revue les principales notions et pratiques d'apprentissage, à partir d'un nombre fini d'observations, de modèles statistiques capables d'effectuer la même tâche sur des données futures. Nous verrons que les algorithmes génériques sont généralement déployés pour apprendre un modèle adapté aux données disponibles. Néanmoins, plusieurs décisions doivent être prises pour s'assurer que les modèles appris (i) s'adaptent aux instances d'apprentissage et (ii) généralisent aux nouvelles instances. Premièrement, les données recueillies doivent être représentatives de la tâche et, parfois, doivent être prétraitées afin de filtrer l'information utile à l'apprentissage ou de réduire la complexité de l'algorithme. Deuxièmement, un algorithme d'apprentissage doit être sélectionné : cela comprend la définition de la classe de fonctions parmi lesquelles le modèle final est choisi pour l'approximation, et au choix de l'algorithme d'optimisation pour la recherche dans l'espace des solutions possibles. De plus, une métrique de performance et une fonction objective sont formulées pour guider l'apprentissage, précisant les caractéristiques souhaitées du modèle final. Grâce à ces mesures, une évaluation empirique du modèle est réalisée tout au long du processus d'apprentissage, à la fois pour guider et contrôler le fitting et aussi pour estimer comment le modèle obtenu fonctionnera sur les données futures. Afin d'obtenir une estimation des capacités réelles de généralisation du modèle appris, l'étude empirique est réalisée en divisant l'échantillon d'apprentissage en deux et en testant le modèle sur le sous-ensemble qui n'est pas utilisé pour l'apprentissage du modèle. Dans un deuxième moment, nous présenterons deux cadres d'étude des capacités de généralisation du modèle appris : les bornes de généralisation Probablement Approximativement Correctes et la Robustesse Adversarielle. Les deux approches reposent sur des considérations indépendantes de la performance empirique du modèle dans l'exécution de la tâche souhaitée. Dans l'ensemble, ils offrent des outils pratiques pour évaluer les propriétés de généralisation du modèle appris et donnent un aperçu du pourquoi ou du pourquoi-pas un modèle qui généraliserait.

Cet aperçu de l'apprentissage statistique ne se veut pas exhaustif : de nombreux concepts ne seront que rapidement cités, d'autres omis, tandis que nous nous attarderons sur tous les éléments nécessaires à la compréhension des apports de cette thèse.

F.2.2 Chapitre 2

Dans le chapitre précédent, nous avons brièvement discuté du lien entre la distribution des données et l'expressivité du modèle. Les caractéristiques des données varient généralement dans l'espace d'entrée : la distribution globale pourrait être multimodale et contenir des non-linéarités. L'algorithme d'apprentissage doit être capable de capturer et de s'adapter à ces changements afin d'avoir de bonnes performances. Même si les modèles linéaires ne parviennent pas à décrire des distributions complexes, ils sont réputés pour leur passage à l'échelle, en entraînement et en test, aux grands ensembles de données en termes de nombre d'exemples et de nombre de caractéristiques. Plusieurs méthodes ont été proposées pour tirer parti du passage à l'échelle et de la simplicité des hypothèses linéaires afin de construire des modèles aux grandes capacités discriminatoires. Ces méthodes améliorent les modèles linéaires, dans le sens qu'elles renforcent leur puissance expressive grâce à différentes techniques. L'un des principaux avantages de ces approches est qu'elles permettent d'éviter la tâche difficile de choisir un modèle approprié à la tâche.

Dans ce chapitre, nous présentons les trois principales approches de cette ligne de recherche. Ces approches intègrent les données dans un nouvel espace de représentation ou s'entraînent et combinent plusieurs modèles sur l'espace d'origine. Nous n'incluons pas une revue complète de la littérature, qui est présentée dans les autres chapitres. Nous donnons plutôt un aperçu des notions, des idées et des principales solutions de cette catégorie d'approches.

Nous commençons par décrire Boosting, un méta-algorithme pour apprendre des modèles forts en utilisant des modèles faibles. Il forme un ensemble d'hypothèses, sur des distributions modifiées de l'ensemble de formation pour assurer leur diversité, et combine leurs prédictions, avec un vote majoritaire pondéré, pour une meilleure performance. Les méthodes de Boosting sont largement déployées pour leur facilité d'application, leur performance et leur fondement théorique.

La deuxième famille que nous allons présenter est la famille des méthodes à noyau, qui apprennent des modèles linéaires sur les espaces latents induits par une fonction noyau sélectionnée. Dans de tels espaces, le produit scalaire entre les vecteurs est donné par la fonction du noyau choisie. Par conséquent, il n'est pas nécessaire de définir explicitement l'espace latent et sa transformation. Nous verrons que les modèles peuvent être formés en utilisant la matrice du noyau évalué pour toutes les paires de points de l'ensemble d'entraînement, appelée matrice de Gram. Les méthodes à noyau sont connues pour leur fondement théorique et leur superbe puissance expressive. Pourtant, leur applicabilité est parfois limitée par leur coût de calcul et de stockage. Il existe plusieurs solutions pour mettre à l'échelle les méthodes à noyau sur de gros ensembles de données. La plupart d'entre elles proposent des approximations de la matrice de Gram (dont la manipulation est souvent un goulet d'étranglement), afin de réduire les coûts d'inversion et de limiter le nombre d'évaluations des noyaux.

Enfin, nous décrivons la famille d'apprentissage localement linéaire, à laquelle appartiennent les contributions de ce manuscrit. Les techniques d'apprentissage locales se concentrent sur la capture des caractéristiques locales de l'espace pour construire des modèles expressifs. Le problème est linéarisé selon le principe de la cohérence locale, selon lequel les prévisions doivent être cohérentes pour les points proches. Nous présentons les deux principales solutions pour obtenir des approximations linéaires du problème. Soit nous partitionnons les données et apprenons un modèle linéaire par sous-ensemble de données, soit nous construisons un espace latent en comparant les instances de l'ensemble des données et un ensemble de points précédemment sélectionnés, répartis sur l'espace d'entrée. Nous montrerons que les approches d'apprentissage locales ont un meilleur passage à l'échelle que les méthodes à noyau.

F.2.3 Chapitre 3

Dans ce chapitre, nous décrivons une méthode pour surmonter les problèmes liés à l'apprentissage local basé sur le partitionnement des données. Grâce à des combinaisons convexes de modèles locaux régularisés en fonction des caractéristiques topologiques de l'espace d'entrée, nous améliorons les techniques d'apprentissage locales typiques. Ce faisant, nous obtenons des prévisions plus continues, nous évitons le sur-apprentissage et nous améliorons les capacités discriminatoires des modèles finaux. Nous concentrons notre travail sur l'apprentissage métrique, un cadre permettant d'améliorer de nombreuses approches d'apprentissage machine en optimisant les distances ou les similarités pour la tâche à accomplir afin qu'elles reflètent les spécificités des données. Dans ce domaine, l'apprentissage métrique local s'est déjà révélé très efficace, notamment pour prendre en compte les non-linéarités dans les données. Cependant, il est bien connu que l'apprentissage métrique local (i) peut entraîner un sur-apprentissage et (ii) rencontre des difficultés pour comparer deux points qui sont assignés à deux modèles locaux différents.

Dans ce chapitre, nous abordons ces deux questions en introduisant un nouvel algorithme d'apprentissage métrique qui combine linéairement des modèles locaux. A partir d'une partition de l'espace en régions et d'un modèle (une fonction de score) pour chaque région, nous abordons ces questions en définissant une métrique entre points comme une combinaison pondérée des modèles locaux. Un vecteur de poids est appris pour chaque paire de régions, et une régularisation spatiale est introduite pour s'assurer que les vecteurs de poids évoluent en douceur et que les modèles proches sont favorisés dans la combinaison. L'approche proposée, appelée Combinaisons convexes de modèles locaux (**C2LM**), a la particularité d'être définie dans un cadre de régression, de travailler implicitement à différentes échelles et d'être suffisamment générique pour être applicable à la fois aux similarités et aux distances.

F.2.4 Chapitre 4

Dans ce chapitre, nous nous concentrons sur l'apprentissage de modèles locaux sur des données partitionnées selon un critère autre que la spatialité. Plus précisément, nous considérons les données générées par un ensemble d'agents, qui ont été collectées par leurs dispositifs personnels afin qu'elles soient naturellement regroupées en sous-ensembles. Comme dans le chapitre précédent, nous visons à apprendre un modèle local par agent (en utilisant son jeu de données local), que nous indiquerons ci-après comme personnalisé. De plus, afin d'améliorer la capacité de généralisation des modèles optimisés, nous ajoutons des contraintes de lissage basées sur un critère de similarité utilisateur. Cependant, contrairement à **C2LM**, nous optimisons tous les modèles personnalisés dans un problème d'optimisation commun en tenant compte à la fois de l'optimalité locale et de la fluidité globale. De plus, l'apprentissage à partir des données personnelles soulève de nouveaux défis que nous devons prendre en compte : les données collectées par chaque utilisateur sont extrêmement volumineuses et potentiellement sensibles. Pour ces raisons, nous envisageons un nouveau paradigme d'apprentissage particulièrement adapté à notre scénario. L'apprentissage décentralisé comprend toutes les techniques travaillant sur un graphe non hiérarchique des utilisateurs, dans lequel chaque utilisateur conserve ses données sur place et ne communique avec ses voisins que les mises à jour du modèle. Notre méthode exploite cette architecture décentralisée pour apprendre des modèles personnalisés de manière collaborative sur un graphe qui reflète les similitudes entre utilisateurs. Comme nous l'illustrerons tout au long de ce chapitre, nous formulons notre problème sous la forme d'un l_1-Adaboost régularisé et l'optimisons en utilisant l'algorithme de Frank-Wolfe, afin d'obtenir des modèles expressifs avec une complexité de communication minimale. Pour pallier l'absence potentielle de connaissances de base sur les similitudes entre les utilisateurs, nous introduisons en outre une formulation pour apprendre conjointement les modèles personnalisés et la topologie du graphe par une procédure d'optimisation alternée.

F.2.5 Chapitre 5

A partir de ce chapitre, nous utilisons un ensemble de points - à savoir des points de repère - répartis sur l'espace d'entrée pour saisir ses caractéristiques locales. Nous allons explicitement former un espace latent en considérant les similitudes entre les entrées et ces repères. Les similarités seront capturées soit par les fonctions noyaux, soit par des approximations linéaires de celles-ci. Ce faisant, nous laissons tomber l'hypothèse que les particularités de la distribution sont constantes pour un sous-ensemble donné de la partition.

En raison de leurs propriétés attrayantes, telles que le passage à l'échelle à de grands ensembles d'entraînement, nous optimisons toujours les modèles linéaires. Cependant, au lieu d'apprendre un modèle local par sous-ensemble de données, nous apprenons un modèle unique pour l'ensemble de l'échantillon sur un espace latent globalement adapté à la tâche en cours. Plus précisément, nous introduisons une adaptation locale de la célèbre méthode des Support Vector Machines (SVMs), que nous appelons **L^3-SVMs**. Le principal défi sera de tirer parti du pouvoir discriminatoire des SVMs à noyau tout en les adaptant à de grands ensembles de données.

Simple et efficace, notre algorithme est aussi théoriquement fondé. En utilisant le cadre de la stabilité uniforme, nous montrons que notre formulation SVM s'accompagne de garanties de généralisation sur le risque réel. Les expériences basées sur la configuration la plus simple de notre modèle (i.e. points de repère choisis au hasard dans l'ensemble de formation, projection linéaire, noyau linéaire) montrent que **L^3-SVMs** est très compétitif par rapport à l'état de l'art et ouvre la porte à de nouvelles lignes de recherche passionnantes.

F.2.6 Chapitre 6

Dans ce dernier chapitre de contributions, nous introduisons une méthode rapide et théoriquement fondée pour apprendre les SVMs à points de repère (**L^3-SVMs**) dans un cadre de classification multi-vues. Dans un tel scénario, les instances de l'ensemble de données sont observées dans plusieurs espaces de caractéristiques. Nous soutenons que la clé pour s'attaquer efficacement aux problèmes multi-vues est d'exploiter la diversité entre vues, car les différentes vues contiennent rarement, à elles seules, suffisamment d'informations pour la tâche à accomplir. La méthode proposée – nommée **MVL-SVM** – tire parti des informations complémentaires des différentes sources d'information et s'adapte linéairement à la taille de l'ensemble de données.

L'approche applique une projection non linéaire à l'ensemble de données au moyen d'estimations de similarité multi-vues par rapport à un ensemble de points de repère sélectionné, avant d'apprendre un SVM linéaire dans l'espace latent reliant toutes les vues. Dans ce nouveau paramètre, les instances de l'échantillon et les repères sont observés dans de multiples espaces de caractéristiques. Nous tenons compte de cette représentation à multiples facettes dans les similitudes entre points et repères : un noyau doit être sélectionné par vue et les similitudes doivent être calculées une vue à la fois (pour comparer un point à un repère, les caractéristiques d'une vue sont comparées aux caractéristiques du repère sur la même vue, et ainsi de suite pour toutes les vues). Dans l'étape finale, en concaténant les représentations obtenues, on parvient à obtenir un espace latent unique, commun à toutes les vues.

A la lumière des résultats empiriques du chapitre 5, l'ensemble des points de repère est choisi au hasard dans l'échantillon d'entraînement. De plus, malgré l'efficacité du couplage du produit scalaire avec le partitionnement des données pour capturer la distribution des données, et dans le but de simplifier la formulation du problème, qui a été surchargé par la complexité accrue de la représentation des données, dans le travail suivant, nous comptons uniquement sur le choix du noyau pour représenter les caractéristiques de l'espace et nous ne faisons aucun clustering des données.

Nous prouvons les capacités de généralisation de notre algorithme à l'aide du cadre de la stabilité uniforme : nous analysons d'abord la robustesse de l'approche face à de légers changements dans l'ensemble d'apprentissage, puis nous en déduisons une limite de généralisation étroite en fonction du nombre de vues et de repères. Dans un second temps, nous étendons notre méthode au scénario de la vue manquante. Grâce à la formulation de notre problème, nous montrons que nous pouvons reconstruire les vues manquantes des points incomplets en estimant simplement les similitudes avec les points de repère dans les vues simples sans approximer les valeurs manquantes des éléments. Les résultats empiriques, qu'il s'agisse du cadre des vues completes ou manquantes, mettent en évidence les performances supérieures de notre méthode, en termes de précision et de temps d'exécution, par rapport à l'état de l'art.

F.3 Conclusions

Dans ce manuscrit, nous avons abordé le problème de l'apprentissage de modèles capables de saisir les caractéristiques locales de la distribution des données, en mettant l'accent sur leur généralisation, leur continuité en prédiction et leur passage à l'échelle. Nous avons contribué sur deux axes de recherche parallèles dans l'apprentissage local : nous avons proposé deux approches basées sur le partitionnement des données et un algorithme basé sur les similitudes avec des points de repère, utilisable sur des données à vue unique et à vues multiples. Des études approfondies ont été menées pour mettre en évidence l'efficacité de ces contributions qui ont confirmé le bien-fondé de leurs intuitions. Nous avons étudié empiriquement les performances des méthodes proposées tant sur des données synthétiques que sur des tâches réelles, en termes de précision et de temps d'exécution, et les avons comparées aux résultats de l'état de l'art. Nous avons également analysé nos approches d'un point de vue théorique, en étudiant leurs complexités de calcul et de mémoire et en dérivant des bornes de généralisation serrées.

Outre les perspectives propres à chacune des contributions que nous avons évoquées dans les chapitres respectifs, nous envisageons plusieurs pistes de recherche supplémentaires. En ce qui concerne le terme de régularisation utilisé dans les chapitres 3 et 4 sur les similitudes entre paires de modèles locaux appris, nous réfléchissons à de nouvelles façons d'estimer le graphe de similitude, i.e. les poids de pénalisation affectés à chaque paire de modèles. Il serait intéressant d'optimiser le graphe comme dans le Chapitre 4, mais de supprimer la contrainte d'uniformité sur les degrés et le nombre de voisins de chaque noeud. Une formulation moins contraignante aurait l'avantage de mieux saisir les relations entre les modèles locaux et de détecter les communautés d'utilisateurs lorsque l'on travaille avec des données personnelles.

Une autre perspective intéressante serait d'étudier les travaux de Partie III sous le cadre de similarités $(\epsilon, \gamma, \tau)-$good présenté au Chapitre 2. En effet, ce cadre offre un moyen bien fondé d'étudier la qualité de la linéarisation du problème induite par les similarités avec les points de repère, comme dans nos travaux. Cependant, il ne serait pas simple d'analyser théoriquement nos contributions à travers cette théorie, car elle ne considère que des points de repère tirés i.i.d. de la distribution des données et que des fonctions de similarité bornées. Ces hypothèses ne sont pas nécessairement satisfaites dans nos travaux : nous admettons l'utilisation de points de repère virtuels et des noyaux non-bornées tels que le noyau linéaire. De plus, nous croyons que le fait de restreindre la fonction de similarité à un noyau de Mercer devrait mener à de meilleurs résultats théoriques.

En outre, nous nous intéressons particulièrement aux méthodes permettant de dessiner de bons ensembles de repères, dans le sens où ils seraient en nombre minimal mais représentatifs de la distribution des données, et d'étudier le compromis entre précision et passage à l'échelle, en fonction de la complexité de la tâche. Nous soutenons qu'une telle étude sur les effets de la qualité de l'échantillon peut conduire à des bornes de généralisation plus serrées.

Enfin, il serait interéssant d'évaluer la robustesse aux exemples adversarial de nos méthodes, en utilisant les mesures rapportées dans le chapitre 1. En fait, nous sommes intrigués par les meilleures robustesses observées dans d'autres modèles qui apprennent des espaces latents par rapport aux modèles qui ne le font pas.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Noga Alon, Shai Ben-David, Nicolo Cesa-Bianchi, and David Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM (JACM)*, 44(4):615–631, 1997.

[2] Massih Amini, Nicolas Usunier, and Cyril Goutte. Learning from multiple partially observed views-an application to multilingual text categorization. In *Advances in neural information processing systems*, pages 28–36, 2009.

[3] Dana Angluin and Philip Laird. Learning from noisy examples. volume 2, pages 343–370. Springer, 1988.

[4] Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in neural information processing systems*, 2015.

[5] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[6] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 2004.

[7] Gökhan Bakır, Léon Bottou, and Jason Weston. Breaking svm complexity with cross training. volume 17, pages 81–88, 2005.

[8] Maria Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. In *COLT*, 2012.

[9] Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. Improved guarantees for learning via similarity functions. *Computer Science Department*, page 126, 2008.

[10] Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. A theory of learning with similarity functions. *Machine Learning*, 72(1-2):89–112, 2008.

[11] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, pages 16–25, New York, NY, USA, 2006. ACM.

[12] Inci M. Baytas, Ming Yan, Anil K. Jain, and Jiayu Zhou. Asynchronous Multi-task Learning. In *ICDM*, 2016.

[13] Ron Bekkerman, Mikhail Bilenko, and John Langford. *Scaling up machine learning: Parallel and distributed approaches.* Cambridge University Press, 2011.

[14] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. Personalized and Private Peer-to-Peer Machine Learning. In *AISTATS*, 2018.

[15] Aurélien Bellet and Amaury Habrard. Robustness and Generalization for Metric Learning. volume 151, pages 259–267, 2015.

[16] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. In *arXiv*, 2013.

[17] Aurélien Bellet, Amaury Habrard, and Marc Sebban. Metric learning. volume 9, pages 1–151. Morgan & Claypool Publishers, 2015.

[18] Aurélien Bellet, Yingyu Liang, Alireza Bagheri Garakani, Maria-Florina Balcan, and Fei Sha. A distributed frank-wolfe algorithm for communication-efficient sparse learning. In *SDM*, 2015.

[19] Shai Ben-David, David Loker, Nathan Srebro, and Karthik Sridharan. Minimizing the misclassification error rate using a surrogate convex loss. In *Proceedings of the 29th International Conference on Machine Learning, ICML*, 2012.

[20] Abhijit Bendale and Terrance E. Boult. Towards open set deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1563–1572, 2016.

[21] Sahely Bhadra, Samuel Kaski, and Juho Rousu. Multi-view kernel completion. volume 106, pages 713–739. Springer, 2017.

[22] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.

[23] Battista Biggio, Igino Corona, Blaine Nelson, Benjamin I. P. Rubinstein, Davide Maiorca, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. Security evaluation of support vector machines in adversarial environments. In *Support Vector Machines Applications*, pages 105–153. Springer, 2014.

[24] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. In *IEEE Transactions on Knowledge and Data Engineering*, volume 26, pages 984–996, 2014.

[25] Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 11. ACM, 2004.

[26] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[27] Antoine Bordes, Léon Bottou, and Patrick Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. volume 10, pages 1737–1754, 2009.

[28] 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. ACM, 1992.

[29] Stéphane Boucheron, Gábor Lugosi, and Olivier Bousquet. Concentration inequalities. In *Advanced Lectures on Machine Learning*, pages 208–240. Springer, 2004.

[30] Olivier Bousquet and André Elisseeff. Stability and generalization. volume 2, pages 499–526. JMLR. org, 2002.

[31] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. volume 3, pages 1–122, 2011.

[32] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[33] Stephen P. Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. volume 52, pages 2508–2530, 2006.

[34] David S. Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[35] Lorenzo Bruzzone, Mingmin Chi, and Mattia Marconcini. A novel transductive svm for semisupervised classification of remote-sensing images. volume 44, pages 3363–3373. IEEE, 2006.

[36] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. volume abs/1607.04311, 2016.

[37] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. volume abs/1705.07263, 2017.

[38] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.

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

[40] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. volume 2, page 27. ACM, 2011.

[41] Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Multi-task learning for boosting with application to web search ranking. In *KDD*, 2010.

[42] Mickaël Chen and Ludovic Denoyer. Multi-view generative adversarial networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 175–188. Springer, 2017.

[43] Mickaël Chen, Ludovic Denoyer, and Thierry Artières. Multi-view data generation without view supervision. 2017.

[44] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

[45] Igor Colin, Aurélien Bellet, Joseph Salmon, and Stéphan Clémençon. Gossip dual averaging for decentralized optimization of pairwise functions. In *ICML*, 2016.

[46] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. volume 48, pages 253–285. Springer, 2002.

[47] Jeff Cooper and Lev Reyzin. Improved algorithms for distributed boosting. In *Allerton*, 2017.

[48] Corinna Cortes and Mehryar Mohri. Auc optimization vs. error rate minimization. In *Advances in neural information processing systems*, pages 313–320, 2004.

[49] Corinna Cortes and Vladimir Vapnik. Support-vector networks. volume 20, pages 273–297. Springer, 1995.

[50] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. volume 13, pages 21–27. IEEE, 1967.

[51] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, KDD '04, pages 99–108, New York, NY, USA, 2004. ACM.

[52] 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*, pages 209–216. ACM, 2007.

[53] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[54] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. volume 89, pages 31–71. Elsevier, 1997.

[55] Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *Control Conference (ECC), 2013 European*, pages 3071–3076. IEEE, 2013.

[56] Petros Drineas and Michael W Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(Dec):2153–2175, 2005.

[57] John C. Duchi, Alekh Agarwal, and Martin J. Wainwright. Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling. volume 57, pages 592–606, 2012.

[58] Cynthia Dwork. Differential Privacy. In *ICALP*, volume 2, 2006.

[59] Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273, 2004.

[60] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. volume 9, pages 1871–1874, 2008.

[61] Jason Farquhar, David Hardoon, Hongying Meng, John S. Shawe-taylor, and Sandor Szedmak. Two view learning: Svm-2k, theory and practice. In *Advances in neural information processing systems*, pages 355–362, 2006.

[62] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting adversarial samples from artifacts. volume abs/1703.00410, 2017.

[63] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pages 2962–2970, 2015.

[64] Marco Fornoni, Barbara Caputo, and Francesco Orabona. Multiclass latent locally linear support vector machines. In *ACML*, pages 229–244, 2013.

[65] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. volume 3, pages 95–110, 1956.

[66] Maurice Fréchet. Sur les fonctionnelles continues. In *Annales Scientifiques de L'Ecole Normale Superieure*, volume 27, pages 193–216, 1910.

[67] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[68] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. volume 55, pages 119–139, 1997.

[69] Yoav Freund, Robert E. Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996.

[70] Jerome H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[71] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. volume 29, pages 1189–1232, 2001.

[72] Andrea Frome, Yoram Singer, and Jitendra Malik. Image retrieval and classification using local distance functions. In *Advances in neural information processing systems*, pages 417–424, 2007.

[73] Zhouyu Fu, Antonio Robles-Kelly, and Jun Zhou. Mixing linear svms for nonlinear classification. volume 21, pages 1963–1975. IEEE, 2010.

[74] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[75] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. volume 12, pages 2211–2268, 2011.

[76] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. volume abs/1704.04960, 2017.

[77] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. volume abs/1412.6572, 2014.

[78] Anil Goyal, Emilie Morvant, Pascal Germain, and Massih-Reza Amini. Pac-bayesian analysis for a two-step hierarchical multiview learning approach. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 205–221. Springer, 2017.

[79] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick D. McDaniel. On the (statistical) detection of adversarial examples. volume abs/1702.06280, 2017.

[80] Quanquan Gu and Jiawei Han. Clustered support vector machines. In *AISTATS*, pages 307–315, 2013.

[81] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. volume 53, pages 217–288. SIAM, 2011.

[82] Trevor Hastie and Robert Tibshirani. Discriminant adaptive nearest neighbor classification and regression. In *Advances in Neural Information Processing Systems*, pages 409–415, 1996.

[83] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *Unsupervised learning*. Springer, 2009.

[84] Søren Hauberg, Oren Freifeld, and Michael J. Black. A geometric take on metric learning. In *Advances in Neural Information Processing Systems*, pages 2024–2032, 2012.

[85] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. volume abs/1512.03385, 2015.

[86] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. volume abs/1610.06940, 2016.

[87] Yinjie Huang, Cong Li, Michael Georgiopoulos, and Georgios C. Anagnostopoulos. Reduced-rank local distance metric learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 224–239. Springer, 2013.

[88] Riikka Huusari, Hachem Kadri, and Cécile Capponi. Multi-view metric learning in vector-valued kernel spaces. In *Proceedings of the Twenty-First*

International Conference on Artificial Intelligence and Statistics, pages 415–424, 2018.

[89] Martin Jaggi. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. In *ICML*, 2013.

[90] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[91] Armand Joulin and Francis R. Bach. A convex relaxation for weakly supervised classifiers. 2012.

[92] Hachem Kadri, Stéphane Ayache, Cécile Capponi, Sokol Koço, François-Xavier Dupé, and Emilie Morvant. The multi-task learning view of multimodal data. In *Asian Conference on Machine Learning*, pages 261–276, 2013.

[93] Purushottam Kar and Prateek Jain. Similarity-based learning via data driven embeddings. In *Advances in neural information processing systems*, pages 1998–2006, 2011.

[94] Michael Kearns and Ming Li. Learning in the presence of malicious errors. volume 22, pages 807–837. SIAM, 1993.

[95] Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 459–468. ACM, 1996.

[96] Michael J. Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[97] Vladimir Koltchinskii and Dmitriy Panchenko. Rademacher processes and bounding the risk of function learning. In *High dimensional probability II*, pages 443–457. Springer, 2000.

[98] Jakub Konečnỳ, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. 2016.

[99] Jakub Konečnỳ, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. 2016.

[100] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 2009.

[101] David Krueger, Nicolas Ballas, Stanislaw Jastrzebski, Devansh Arpit, Maxinder S. Kanwal, Tegan Maharaj, Emmanuel Bengio, Asja Fischer, Aaron Courville, Simon Lacoste-Julien, and Yoshua Bengio. A closer look at memorization in deep networks. In *International Conference on Machine Learning (ICML)*, 2017. arxiv:1706.05394.

[102] Brian Kulis. Metric learning: A survey. volume 5, pages 287–364, 2012.

[103] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. volume abs/1607.02533, 2016.

[104] Matt Kusner, Stephen Tyree, Kilian Q Weinberger, and Kunal Agrawal. Stochastic neighbor compression. In *Proceedings of the 31st international conference on machine learning (ICML-14)*, pages 622–630, 2014.

[105] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pages 496–504, 2015.

[106] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-Coordinate Frank-Wolfe Optimization for Structural SVMs. In *ICML*, 2013.

[107] Lubor Ladicky and Philip Torr. Locally linear support vector machines. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 985–992, 2011.

[108] Jean Lafond, Hoi-To Wai, and Eric Moulines. D-FW: Communication efficient distributed algorithms for high-dimensional sparse optimization. In *ICASSP*, 2016.

[109] Rémi Lajugie, Damien Garreau, Francis R. Bach, and Sylvain Arlot. Metric learning for temporal sequence alignment. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 1817–1825, 2014.

[110] Aleksandar Lazarevic and Zoran Obradovic. The distributed boosting algorithm. In *KDD*, 2001.

[111] Rémi Lebret and Ronan Collobert. Word emdeddings through hellinger pca. 2013.

[112] Yann LeCun and Yoshua Bengio. The handbook of brain theory and neural networks. chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. MIT Press, Cambridge, MA, USA, 1998.

[113] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. volume 521, pages 436–444. Nature Research, 2015.

[114] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. volume 86, pages 2278–2324, Nov 1998.

[115] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[116] Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Julie Beaulieu, Peter J. Bentley, Samuel Bernard, Guillaume Belson, David M Bryson, Nick Cheney, et al. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *arXiv preprint arXiv:1803.03453*, 2018.

[117] Jiyi Li, Tomohiro Arai, Yukino Baba, Hisashi Kashima, and Shotaro Miwa. Distributed Multi-task Learning for Sensor Network. In *ECML/PKDD*, 2017.

[118] Yu-Feng Li, Ivor W. Tsang, James T. Kwok, and Zhi-Hua Zhou. Convex and scalable weakly labeled svms. volume 14, pages 2151–2188. JMLR. org, 2013.

[119] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In *NIPS*, 2017.

[120] M. Lichman. UCI machine learning repository, 2013.

[121] Shan Sung Liew, Mohamed Khalil-Hani, and Rabia Bakhteri. Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems. volume 216, pages 718–734. Elsevier, 2016.

[122] Lydia T. Liu, Sarah Dean, Esther Rolf, Max Simchowitz, and Moritz Hardt. Delayed impact of fair machine learning. *ICML*, 2018.

[123] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pages 2370–2378, 2016.

[124] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[125] Lynn H. Loomis. *Introduction to abstract harmonic analysis.* Courier Corporation, 2013.

[126] Daniel Lowd and Christopher Meek. Adversarial learning. In *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD)*, KDD '05, pages 641–647, New York, NY, USA, 2005. ACM.

[127] Chenxin Ma, Jakub Konečnỳ, Martin Jaggi, Virginia Smith, Michael I Jordan, Peter Richtárik, and Martin Takáč. Distributed optimization with arbitrary local solvers. volume 32, pages 813–848, 2017.

[128] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. volume abs/1706.06083, 2017.

[129] Yishay Mansour and Mariano Schain. Robust domain adaptation. volume 71, pages 365–380. Springer, 2014.

[130] Llew Mason, Jonathan Baxter, Peter L. Bartlett, and Marcus R. Frean. Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pages 512–518, 2000.

[131] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[132] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. volume abs/1702.04267, 2017.

[133] Charles A. Micchelli and Massimiliano Pontil. On learning vector-valued functions. volume 17, pages 177–204. MIT Press, 2005.

[134] Ha Quang Minh, Loris Bazzani, and Vittorio Murino. A unifying framework for vector-valued manifold regularization and multi-view learning. In *ICML (2)*, pages 100–108, 2013.

[135] Ha Quang Minh, Loris Bazzani, and Vittorio Murino. A unifying framework in vector-valued reproducing kernel hilbert spaces for manifold regularization and co-regularized multi-view learning. volume 17, pages 1–72, 2016.

[136] Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[137] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. volume abs/1610.08401, 2016.

[138] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. Analysis of universal adversarial perturbations. volume abs/1705.09554, 2017.

[139] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. volume abs/1511.04599, 2015.

[140] Emilie Morvant, Amaury Habrard, and Stéphane Ayache. Parsimonious unsupervised and semi-supervised domain adaptation with good similarity functions. volume 33, pages 309–349. Springer, 2012.

[141] Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep K. Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[142] Angelia Nedic and Asuman E. Ozdaglar. Distributed Subgradient Methods for Multi-Agent Optimization. volume 54, pages 48–61, 2009.

[143] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. volume abs/1412.1897, 2014.

[144] Maria-Irina Nicolae, Marc Sebban, Amaury Habrard, Éric Gaussier, and Massih-Reza Amini. Algorithmic robustness for semi-supervised (ϵ, γ, τ)-good metric learning. 2014.

[145] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Ian M Molloy, and Ben Edwards. Adversarial robustness toolbox v0. 2.2. *arXiv preprint arXiv:1807.01069*, 2018.

[146] Richard Nock and Frank Nielsen. Bregman divergences and surrogates for learning. volume 31, pages 2048–2059. IEEE, 2009.

[147] Yung-Kyun Noh, Byoung-Tak Zhang, and Daniel D. Lee. Generative local metric learning for nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 1822–1830, 2010.

[148] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. 2015.

[149] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer*

and Communications Security, ASIA CCS '17, pages 506–519, New York, NY, USA, 2017. ACM.

[150] Nicolas Papernot and Patrick D. McDaniel. On the effectiveness of defensive distillation. volume abs/1607.05113, 2016.

[151] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. volume abs/1511.07528, 2015.

[152] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. volume abs/1511.04508, 2015.

[153] Giorgio Patrini, Frank Nielsen, Richard Nock, and Marcello Carioni. Loss factorization, weakly supervised learning and label noise robustness. 2016.

[154] Giorgio Patrini, Richard Nock, Tiberio Caetano, and Paul Rivera. (almost) no label no cry. In *Advances in Neural Information Processing Systems*, pages 190–198, 2014.

[155] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. volume 12, pages 2825–2830, 2011.

[156] Michaël Perrot, Amaury Habrard, Damien Muselet, and Marc Sebban. Modeling perceptual color differences by local metric learning. In *European conference on computer vision*, pages 96–111. Springer, 2014.

[157] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[158] Carl Edward Rasmussen and Christopher KI Williams. Gaussian processes for machine learning. 2006. *The MIT Press, Cambridge, MA, USA*, 38:715–719, 2006.

[159] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? volume 16, pages 1063–1076. MIT Press, 2004.

[160] Amir Rosenfeld and John K. Tsotsos. Intriguing properties of randomly weighted networks: Generalizing while learning next to nothing. *arXiv preprint arXiv:1802.00844*, 2018.

[161] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[162] Robert E. Schapire, Yoav Freund, Peter Bartlett, Wee Sun Lee, et al. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.

[163] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

[164] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[165] Bernhard Schölkopf, Alexander J. Smola, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[166] Ohad Shamir and Nathan Srebro. Distributed Stochastic Optimization and Learning. In *Allerton*, 2014.

[167] Gaurav Sharma, Wencheng Wu, and Edul N. Dalal. The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. volume 30, pages 21–30. New York: Wiley, 1976-, 2005.

[168] Chunhua Shen and Hanxi Li. On the dual formulation of boosting algorithms. volume 32, pages 2216–2231, 2010.

[169] Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622. ACM, 2008.

[170] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[171] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. Federated Multi-Task Learning. In *NIPS*, 2017.

[172] Yangqiu Song, Zhengdong Lu, Cane Wing-ki Leung, and Qiang Yang. Collaborative boosting for activity classification in microblogs. In *KDD*, 2013.

[173] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. volume 15, pages 1929–1958. JMLR.org, January 2014.

[174] Ingo Steinwart. Sparseness of support vector machines. volume 4, pages 1071–1105, 2003.

[175] Shiliang Sun. Multi-view laplacian support vector machines. In *International Conference on Advanced Data Mining and Applications*, pages 209–222. Springer, 2011.

[176] Shiliang Sun. A survey of multi-view machine learning. volume 23, pages 2031–2038. Springer, 2013.

[177] Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and H. Brendan McMahan. Distributed Mean Estimation with Limited Communication. In *ICML*, 2017.

[178] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. 2011.

[179] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. volume abs/1312.6199, 2013.

[180] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D^2: Decentralized training over decentralized data. 2018.

[181] Hanlin Tang, Ce Zhang, Shaoduo Gan, Tong Zhang, and Ji Liu. Decentralization Meets Quantization. 2018.

[182] Jingjing Tang, Yingjie Tian, Peng Zhang, and Xiaohui Liu. Multiview privileged support vector machines. IEEE, 2017.

[183] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855. ACM, 2013.

[184] Marko Tkalcic, Jurij F. Tasic, et al. Colour spaces: perceptual, historical and applicational background. In *Eurocon*, pages 304–308, 2003.

[185] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. 2017.

[186] Anusua Trivedi, Piyush Rai, Hal Daumé III, and Scott L. DuVall. Multiview clustering with incomplete views. In *NIPS Workshop*, 2010.

[187] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. There is no free lunch in adversarial robustness (but there are unexpected benefits). *arXiv preprint arXiv:1805.12152*, 2018.

[188] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[189] Aad W. Van Der Vaart and Jon A. Wellner. Weak convergence. In *Weak Convergence and Empirical Processes*, pages 16–28. Springer New York, 1996.

[190] Jan C. Van Gemert, Jan-Mark Geusebroek, Cor J. Veenman, and Arnold W. M. Smeulders. Kernel codebooks for scene categorization. In *European conference on computer vision*, pages 696–709. Springer, 2008.

[191] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. Decentralized Collaborative Learning of Personalized Models over Networks. In *AISTATS*, 2017.

[192] Vladimir Vapnik. *Estimation of dependences based on empirical data*. Springer Science & Business Media, 2006.

[193] Vladimir N. Vapnik and A. Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.

[194] Manik Varma. Local deep kernel learning for efficient non-linear svm prediction. In *International Conference on Machine Learning*, pages 486–494, 2013.

[195] Chu Wang, Yingfei Wang, Robert Schapire, et al. Functional Frank-Wolfe Boosting for General Loss Functions. 2015.

[196] Jialei Wang, Mladen Kolar, and Nathan Srebro. Distributed Multi-Task Learning with Shared Representation. 2016.

[197] Jialei Wang, Mladen Kolar, and Nathan Srebro. Distributed Multitask Learning. In *AISTATS*, 2016.

[198] Joseph Wang and Venkatesh Saligrama. Local supervised learning through space partitioning. In *Advances in Neural Information Processing Systems*, pages 91–99, 2012.

[199] Jun Wang, Alexandros Kalousis, and Adam Woznica. Parametric local metric learning for nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 1601–1609, 2012.

[200] Shijun Wang and Changshui Zhang. Network game and boosting. In *ECML*, 2005.

[201] David Warde-Farley and Ian Goodfellow. Adversarial perturbations of deep neural networks. In Tamir Hazan, George Papandreou, and Daniel Tarlow, editors, *Perturbation, Optimization, and Statistics*. 2016.

[202] Ermin Wei and Asuman E. Ozdaglar. Distributed Alternating Direction Method of Multipliers. In *CDC*, 2012.

[203] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2005.

[204] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. volume 10, pages 207–244. JMLR. org, 2009.

[205] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018.

[206] Christopher K. I. Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

[207] Eric P. Xing, Michael I. Jordan, Stuart Russell, and Andrew Y. Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002.

[208] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. volume abs/1304.5634, 2013.

[209] Huan Xu and Shie Mannor. Robustness and generalization. volume 86, pages 391–423. Springer, 2012.

[210] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. volume abs/1704.01155, 2017.

[211] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing mitigates and detects carlini/wagner adversarial examples. volume abs/1705.10686, 2017.

[212] Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. volume 2, 2006.

[213] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2223–2231. Curran Associates, Inc., 2009.

[214] Eliezer Yudkowsky. Artificial intelligence as a positive and negative factor in global risk. *Global catastrophic risks*, 1(303):184, 2008.

[215] Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. Decentralized Frank-Wolfe boosting for collaborative learning of personalized models. In *CAp*, 2018.

[216] Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. Apprentissage de combinaisons convexes de métriques locales avec garanties de généralisation. In *CAp2016*, 2016.

[217] Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. Beta-risk: a new surrogate risk for learning from weakly labeled data. In *Advances in Neural Information Processing Systems*, pages 4365–4373, 2016.

[218] Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. Lipschitz continuity of mahalanobis distances and bilinear forms. 2016.

[219] Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. Metric learning as convex combinations of local models with generalization guarantees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1478–1486, 2016.

[220] Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. L ^{3}-svms: Landmark-based linear local support vector machines. In *CAp*, 2017.

[221] Valentina Zantedeschi, Rémi Emonet, and Marc Sebban. Fast and provably effective multi-view classification with landmark-based svm. In *ECML PKDD*, 2018.

[222] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 39–49. ACM, 2017.

[223] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. volume abs/1611.03530, 2016.

[224] Chunlei Zhang, Muaz Ahmad, and Yongqiang Wang. Admm based privacy-preserving decentralized optimization. IEEE, 2018.

[225] Yuchen Zhang, John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *NIPS*, 2013.

[226] Yuchen Zhang, Martin J. Wainwright, and John C. Duchi. Communication-efficient algorithms for statistical optimization. In *NIPS*, 2012.

[227] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. volume 38, pages 43–54. Elsevier, 2017.

[228] Xi Zhou, Na Cui, Zhen Li, Feng Liang, and Thomas S. Huang. Hierarchical gaussianization for image classification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1971–1977. IEEE, 2009.

[229] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Abstract In Machine Learning field, data characteristics usually vary over the space: the overall distribution might be multi-modal and contain non-linearities. In order to achieve good performance, the learning algorithm should then be able to capture and adapt to these changes. Even though linear models fail to describe complex distributions, they are renowned for their scalability, at training and at testing, to datasets big in terms of number of examples and of number of features. Several methods have been proposed to take advantage of the scalability and the simplicity of linear hypotheses to build models with great discriminatory capabilities. These methods empower linear models, in the sense that they enhance their expressive power through different techniques. This dissertation focuses on enhancing local learning approaches, a family of techniques that infers models by capturing the local characteristics of the space in which the observations are embedded. The founding assumption of these techniques is that the learned model should behave consistently on examples that are close, implying that its results should also change smoothly over the space. The locality can be defined on spatial criteria (e.g. closeness according to a selected metric) or other provided relations, such as the association to the same category of examples or a shared attribute. Local learning approaches are known to be effective in capturing complex distributions of the data, avoiding to resort to selecting a model specific for the task. However, state of the art techniques suffer from three major drawbacks: they easily memorize the training set, resulting in poor performance on unseen data; their predictions lack of smoothness in particular locations of the space; they scale poorly with the size of the datasets. The contributions of this dissertation investigate the aforementioned pitfalls in two directions: we propose to introduce side information in the problem formulation to enforce smoothness in prediction and attenuate the memorization phenomenon; we provide a new representation for the dataset which takes into account its local specificities and improves scalability. Thorough studies are conducted to highlight the effectiveness of the said contributions which confirmed the soundness of their intuitions. We empirically study the performance of the proposed methods both on toy and real tasks, in terms of accuracy and execution time, and compare it to state of the art results. We also analyze our approaches from a theoretical standpoint, by studying their computational and memory complexities and by deriving tight generalization bounds.

Résumé Dans le domaine de l'apprentissage machine, les caractéristiques des données varient généralement dans l'espace des entrées : la distribution globale pourrait être multimodale et contenir des non-linéarités. Afin d'obtenir de bonnes performances, l'algorithme d'apprentissage devrait alors être capable de capturer et de s'adapter à ces changements. Même si les modèles linéaires ne parviennent pas à décrire des distributions complexes, ils sont réputés pour leur passage à l'échelle, en entraînement et en test, aux grands ensembles de données en termes de nombre d'exemples et de nombre de fonctionnalités. Plusieurs méthodes ont été proposées pour tirer parti du passage à l'échelle et de la simplicité des hypothèses linéaires afin de construire des modèles aux grandes capacités discriminatoires. Ces méthodes améliorent les modèles linéaires, dans le sens où elles renforcent leur expressivité grâce à différentes techniques. Cette thèse porte sur l'amélioration des approches d'apprentissage locales, une famille de techniques qui infère des modèles en capturant les caractéristiques locales de l'espace dans lequel les observations sont intégrées. L'hypothèse fondatrice de ces techniques est que le modèle appris doit se comporter de manière cohérente sur des exemples qui sont proches, ce qui implique que ses résultats doivent aussi changer de façon continue dans l'espace des entrées. La localité peut être définie sur la base de critères spatiaux (par exemple, la proximité en fonction d'une métrique choisie) ou d'autres relations fournies, telles que l'association à la même catégorie d'exemples ou un attribut commun. On sait que les approches locales d'apprentissage sont efficaces pour capturer des distributions complexes de données, évitant de recourir à la sélection d'un modèle spécifique pour la tâche. Cependant, les techniques de pointe souffrent de trois inconvénients majeurs : ils mémorisent facilement l'ensemble d'entraînement, ce qui se traduit par des performances médiocres sur de nouvelles données ; leurs prédictions manquent de continuité dans des endroits particuliers de l'espace ; elles évoluent mal avec la taille des ensembles de données. Les contributions de cette thèse examinent les problèmes susmentionnés dans deux directions : nous proposons d'introduire des informations secondaires dans la formulation du problème pour renforcer la continuité de la prédiction et atténuer le phénomène de mémorisation ; nous fournissons une nouvelle représentation de l'ensemble de données qui tient compte de ses spécificités locales et améliore son évolutivité. Des études approfondies sont menées pour mettre en évidence l'efficacité de ces contributions pour confirmer le bien-fondé de leurs intuitions. Nous étudions empiriquement les performances des méthodes proposées tant sur des données synthétiques que sur des tâches réelles, en termes de précision et de temps d'exécution, et les comparons aux résultats de l'état de l'art. Nous analysons également nos approches d'un point de vue théorique, en étudiant leurs complexités de calcul et de mémoire et en dérivant des bornes de généralisation serrées.