Apprentissage de représentation dirigée par la tâche

Pauline Wauquier

École doctorale ED072 "Sciences pour l'Ingénieur"

Task driven representation learning

Thèse préparée par

Pauline WAUQUIER

Pour l'obtention du grade de

Docteur de l'Université de Lille

Domaine: **Informatique**

soutenue le 29 mai 2017

Centre de Recherche en Informatique, Signal et Automatique de Lille
(CRIStAL - UMR CNRS 9189)

Composition du jury:

Denoyer Ludovic	Professeur à l'Université Pierre et Marie Curie	Rapporteur
Fromont Elisa	Maître de conférences HDR en informatique à université de Saint Etienne	Examinatrice
Jourdan Laetitia	Professeur à l'Université de Lille 1	Examinatrice
Viennet Emmanuel	Professeur à l'Université Paris 13	Rapporteur
Keller Mikaela	Maître de conférences à l'Université Lille 3	Examinatrice, Co-directrice
Tommasi Marc	Professeur à l'Université Lille 3	Examinateur, Co-directeur
Grigolato Frédérique	Dirigeante de Clic and Walk	Invitée

THESIS: Learning adaptively a graph allows us to more efficiently solve semi-supervised tasks.

Summary: *Machine learning* propose numerous algorithms to solve the different tasks that can be extracted from real world prediction problems. To solve the different concerned tasks, most *Machine learning* algorithms somehow rely on relationships between instances. Pairwise instances relationships can be obtained by computing a distance between the vectorial representations of the instances. Considering the available vectorial representation of the data, none of the commonly used distances is ensured to be representative of the task that aims at being solved. In this work, we investigate the gain of tuning the vectorial representation of the data to the distance to more optimally solve the task.

We more particularly focus on the graph-based algorithm introduced in [Zhu and Ghahramani, 2002] for classification task. An algorithm to learn a mapping of the data in a representation space which allows an optimal graph-based classification is first introduced. By projecting the data in a representation space in which the predefined distance is representative of the task, we aim at outperforming the initial vectorial representation of the data when solving the task. A theoretical analysis of the introduced algorithm is performed to define the conditions ensuring an optimal classification. A set of empirical experiments allows us to evaluate the gain of the introduced approach and to temper the theoretical analysis.

Résumé: De nombreux algorithmes d'*Apprentissage automatique* ont été proposés afin de résoudre les différentes tâches pouvant être extraites des problèmes de prédiction issus d'un contexte réel. Pour résoudre les différentes tâches pouvant être extraites, la plupart des algorithmes d'*Apprentissage automatique* se basent d'une manière ou d'une autre sur des relations liant les instances. Les relations entre paires d'instances peuvent être définies en calculant une distance entre les représentations vectorielles des instances. En se basant sur la représentation vectorielle des données, aucune des distances parmi celles communément utilisées n'est assurée d'être représentative de la tâche à résoudre. Dans ce document, nous étudions l'intérêt d'adapter la représentation vectorielle des données à la distance utilisée pour une meilleure résolution de la tâche.

Nous nous concentrons plus précisément sur l'algorithme introduit dans [Zhu and Ghahramani, 2002], résolvant une tâche de classification en se basant sur un graphe. Nous décrivons d'abord un algorithme apprenant une projection des données dans un espace de représentation permettant une résolution, basée sur un graphe, optimale de la classification. En projetant les données dans un espace de représentation dans lequel une distance préalablement définie est représentative de la tâche, nous pouvons surpasser la

représentation vectorielle des données lors de la résolution de la tâche. Une analyse théorique de l'algorithme décrit est développée afin de définir les conditions assurant une classification optimale. Un ensemble d'expériences nous permet finalement d'évaluer l'intérêt de l'approche introduite et de nuancer l'analyse théorique.

Contents

1 Introduction **1**

2 Semi-supervised machine learning **8**
2.1 From real world problems to Machine learning tasks 8
2.2 Supervised classification 11
2.3 Semi-supervised classification 12
2.4 Metric learning & representation learning 14

3 Graph-based semi-supervised learning **21**
3.1 Graph representation 21
3.2 Graph construction methods 23
3.3 Semi-supervised graph-based algorithms 28
3.4 Graph construction relevance 35

4 A metric driven representation learning for graph-based label propagation **37**
4.1 Representation learning for graph-based classification 38
4.2 Constrained representation space learning 40
4.3 Non-linear neural network learning 43
4.4 Graph-based classification 50
4.5 Related work 51

5 Bounds on the classification error **53**
5.1 Motivations 53
5.2 Theoretical guarantees 60
5.3 Proof 64

6 Empirical analysis **70**
6.1 MDRL - Triplet based neural network learning 71
6.2 Datasets 72
6.3 Comparative settings 79
6.4 Representation learning 81
6.5 Empirical evaluation of the theoretical assumptions 85
6.6 Label propagation classification evaluation 89

6.7 Do we need a graph? 92

7 Conclusion & future work **97**
7.1 Conclusion 97
7.2 Open questions 98
7.3 *Clic and Walk* case study 101

Chapter 1

Introduction

The thesis that led to the present work was performed in partnership with the Clic and Walk *start-up (CIFRE industrial agreement), which is a company that performs market surveys and collects customers opinions for others companies. Upon request of the companies, missions, composed by a set of questions and pictures, are sent to the* Clic and Walk *users, which get a remuneration in return for their answers. Time and cost constraints have lead to a need for a better policy for missions allocation among the users.*

During the thesis, I had the opportunity to perform theoretical researchs with the aim of deploying a users recommendation within the framework of the company. The following work presents the theoretical researchs that have been performed during the present thesis on task driven representation learning.

There are various real world problems that are solved by predicting an information from a dataset. For illustration purposes, let us consider a commercial website and some associated marketing problems. Let us suppose that once registered, customers can buy and grade the numerous products that are available on the website. Employees of the commercial website have access to the profile of their customers, their ratings and their purchase records. To increase the number of transactions, employees want to encourage customers to buy more items. They hence need to propose customers interesting products to purchase. Interest of a specific customer for an item needs to be predicted based on the available information, i.e. the profile of the customer and of the item. Another marketing improvement is to be able to partition the set of customers for market segmentation, either based on the profile of the customers or on their purchase records. A last problem for commercial websites that will be highlighted is the budget forecast. The amount a customer is prone to spend on the website needs to be predicted, depending on its profile.

Among the different prediction tasks that can be extracted from real

world problems, let us cite:

- **Classification**: the problem is to associate a label with each instance (for example the interest of a customer for an item)
- **Regression**: a continuous value musts be associated with each instance (for example the expense of the customers)
- **Clustering**: the objective is to define a partition of the instances set (for example the market segmentation)

The introduced prediction problems can be solved by asking domain experts to analyse the collected data in order to detect a pattern, to define rules or to extract the required information. A commercial website can for instance employ someone in order to create associations between items and customers information. Manually creating every association is however time and money costly as the website needs someone to treat a growing set of items and customers. Automatically predict the interest a user can have for an item based on the user and item information is a more profitable solution for the commercial website. Considering the expense of the set of customers prediction problem, previous years expenses of each customer jointly to its profile should be analysed by a dedicated employee to predict next year expenses. Being able to automatically predict the amount of money a customer will spend based on his profile, purchase records and other customers would be a better solution than manually solving the task.

Defining rules and discovering pattern manually is not only time consuming and expensive. It can also be too complex for a human to complete. Notably because of the data amount and complexity, some problems are too arduous to be optimally solved by hand. Manually solving the marketing segmentation problem for example requires someone to explore and analyse the behaviour and information of a set of users in order to extract groups. Due to the number of customers and variables associated with each user, market segmentation problem quickly becomes too complex.

Machine learning algorithms propose several approaches in order to answer the several introduced issues. *Machine learning* methods aim at optimally solving real world tasks. They are developed to automatically solve the task based on an input dataset, without requiring review from experts during the learning process.

A dataset is the set of information from which reason *Machine learning* algorithms to solve the task. Datasets are commonly composed of the set of instances the algorithm either learns from or for which it aims at solving the task. Supplementary information are sometimes available during the data collection process. The task related label of an instance can be available

during the data creation process or may have been proposed by an expert. The label that aims at being predicted for a specific task may thus be accessible for some instances. The commercial websites for example has access to the set of items their customers purchased during previous years and can easily compute the money spent by a customer. For the items recommendation problem, interest of customers can be induced for items they already marked or bought, based on their purchase records.

Algorithms can take advantage of the existence of such additional information. Depending on the fact that *Machine learning* algorithms exploit the available additional information or not, they can be grouped in different categories. Let us highlight the following ones:

- **supervised learning**: a label is associated with each instance of the dataset available at training time. Supervised algorithms aim at learning a model that is able to predict a label for any instance generated from the same distribution.
- **semi-supervised learning**: the data from which reason semi-supervised algorithms is composed of set of labeled instances and a set of unlabeled instances. A semi-supervised algorithm exploits the whole set of instances — labeled and unlabeled one — to predict a label for any instances drawn from the same distribution.
- **unsupervised learning**: unsupervised algorithms are trained from a set of unlabeled instances. They indeed learn a model based on the vectorial representation of the whole training set without having any label. The task is solved only based on the distribution and the vectorial representation of the data.

Machine learning algorithms are applied on datasets in order to learn a model or to extract the needed information. Data first needs to be collected from the problem real world context, for the task to be solved. For the commercial website examples, the real world data related to the different problems can for example be the customers, the information collected during their registration process or their purchase records. Most *Machine learning* algorithms expect a vectorial representation of the data. Applying *Machine learning* algorithms requires to define a modelling of the real world data into a description vector.

Considering the problem of predicting the expense of each customer, the registration and purchases records information of the customers can be used to solve the task. The vectorial representation of the data can for example be composed by the age, the gender, the seniority, the hair colour, the previous year expenses or the city the customers live in. The different features that can be extracted from real world data are from different types. The vectorial

representation of a dataset can be composed of continuous and discrete features. For example, the previous year expenses information for a customer is a real positive number and consequently a continuous feature. Other attributes are discrete, for example the age of a customer or the numerical transformation of categorical features like the gender of a customer. Among the several attributes that are available in the real world context, some may be useless in order to solve the task. The hair colour of the customer can for example be neglected to solve the expense prediction task. The set of attributes that will composed the vectorial representation of the dataset musts first be selected. Among the selected features, some are categorical information. In our example, the gender of a customer and its ZIP code are two categorical information. Categorical information are hardly directly usable in a vectorial representation. It consequently needs to be expressed either as a continuous or as a discrete numerical features. An expert can be asked to propose a vectorial representation of the data based on the available information. Numerous vectorial representations of a dataset can be computed based on the several choices made by the expert. The results of an algorithm are dependent on the vectorial representation on which the algorithm is applied and can vary for the different available vectorial representation of the data. The vectorial modelling of the dataset is consequently an important step to solve the targeted task.

Most *Machine learning* algorithms compute a distance or a similarity between the vectorial representations of the data. Several general distances and similarities have been classically defined and are commonly used. For the remainder of this work, let us focus on distances. It should however be pointed that equivalent statements about similarities could be easily reached by inverting the scale. Small (respectively large) distances should be replaced by large (respectively small) similarities.

Classical distances give equal prominence to all the features of the vectorial representation of the data. Because of arbitrary choices during the processes leading to the vectorial representation of the data, all the features of a vectorial representation of an instance do however not have equal influence considering the task to solve. The assumption of equal influence of all features on which distances are based being unsatisfied, any classical distance between the vectorial representation of two instances may thus be sub-optimal considering the target task.

The computation of an algorithm for a real world problem is dependent on the distance applied to the vectorial representation of the data. The capability to solve the task thus depends on the choices made for the vectorial representation and the distance. Depending on the fact that either the representation of the dataset or the distance between instances are prominent for the algorithm, either the representation or the similarities between instances can be learned.

In the literature, a body of work is dedicated to the problem of mutually adapting the vectorial representation and the metric. Numerous approaches have been proposed to either adapt the distance to the selected vectorial representation (*metric learning*) or to adapt the vectorial representation to the set distance (*representation learning*, *features selection*, *dimensionality reduction*).

Once a vectorial representation of the dataset and a distance are set, a graph representation of the dataset can be built to solve the targeted task. A subset of semi-supervised algorithms indeed takes advantage of graph structured datasets to solve the different tasks.

Graphs are composed by a set of vertices which are linked by a set of weighted edges. The weights used to characterize the edges represent a relationship between the vertices linked by the edges. Links between pairs of instances from a dataset may for example be created based on a distance between the vectorial representations of the instances. Several graph representations can be computed from a vectorial dataset.

Graph-based semi-supervised algorithms decisions rely on the relationships between instances. Graph-based algorithms assume that the graph is a homophilic structure, i.e. that close instances in the graph are similar, considering the target task. As previously seen, the vectorial representation of a dataset and the distance may have no adequacy for the task to be solved. The graph representation of a dataset based on the set distance and vectorial representation may not satisfy the smoothness assumption and graph-based algorithms can perform poorly.

The main question underlying this work is related to the method to use to tune at best the vectorial representation of the data and its associated distance. How can the representation space and the distance associated with it be learned such that the distance on the representation space is meaningful considering the task to be solved? Metric and representation learning approaches are interrelated. In this work, considering a set distance, we study representation learning approaches for graph-based resolution of the targeted task. We claim that a task-driven representation learning algorithm allows graph-based semi-supervised algorithms to outperformed the initial representation space of the data.

The outline of the thesis will be as follows:

- *Chapter 2: we introduce the need for metric learning and representation learning for semi-supervised tasks.* After introducing main tasks of *Machine learning*, we present supervised and semi-supervised learning. Some of supervised and semi-supervised learning algorithms are

reviewed. We discuss the influence of the representation space of the instances and the influence of the distance that is used by the algorithm. Some metric learning and representation learning approaches, along with their limitations are then reviewed.

- *Chapter 3: we present graph-based semi-supervised learning.* We first review graphs description, characteristics and properties. Based on a dataset, different graph representation construction methods are introduced. Semi-supervised algorithms exploiting a graph representation of a dataset are then reviewed. We finally discuss the influence of the construction methods on graph-based algorithms computation.

- *Chapter 4:we introduce our metric driven representation learning algorithm for graph-based classification.* For an efficient graph-based algorithm computation, the data aims at being projected in a representation space that satisfies some smoothness related constraints. Based on the constraints the representation space is required to satisfy, the cost function being optimized is introduced. An algorithm to learn a mapping from the initial space to a representation space that satisfies some constraints is described. Due to some of their properties, neural networks are interesting models for representation learning. The neural network function that is optimized and the siamese architecture used to learn our mapping function are then introduced. Finally, we see how the representation space learned by the introduced algorithm is exploited in order to apply a label propagation algorithm on the built graph.

- *Chapter 5: theoretical guarantees on the classification accuracy in our framework are introduced.* We show that if a representation space is learned such that it satisfies a specific set of relative constraints and if the data satisfies some initial distribution constraints, we can build a graph for an optimal graph-based classification. The distance between two instances in the representation space learned by the introduced algorithm can indeed be bounded depending on their distance in the initial space. We define theoretical conditions such that some relative constraints are satisfied in the representation space instances are projected in. We then show that a threshold can be defined in the representation space instances are projected in such that two instances can be linked to each other if and only if they are similarly labeled, allowing to perform an optimal classification.

- *Chapter 6: capabilities of our framework for classification on artificial and real datasets are empirically evaluated.* The introduced representation learning algorithm is driven by a set of constraints. The capability of our algorithm to learn a representation space that respects the spe-

cified constraints is first evaluated, by verifying if the constraints are respected in the representation space and how it generalizes. A theoretical analysis has defined the conditions under which the introduced representation learning algorithm allows an optimal classification. The theoretical analysis being based on two strong assumptions, satisfaction of the assumptions on the several datasets is being empirically evaluated. The gain of introduced representation learning algorithm for the classification task musts itself be empirically evaluated. The gain of the representation learning on the classification accuracy is consequently analysed, by comparing the classification error obtained on a ϵ-graph built in the representation space proposed by the introduced algorithm to the error obtained on graphs obtained with other metric and representation learning algorithms. We finally explore possible extensions of this work by comparing the graph-based classification algorithm with other classification algorithms in the representation space learned by our introduced representation learning algorithm.

- *Chapter 7:* We contrast our claim with the theoretical analysis and the empirical experiments. We discuss some open questions that emerged from exploratory works and some prospects are introduced.

Chapter 2

Semi-supervised machine learning

To optimally and automatically solve some real world problems, *Machine learning* algorithms reason from datasets created from real world context. Labels may be accessible for some instances, leading to have labeled examples. Supervised learning algorithms take advantage of labeled examples to solve the targeted task. After a more precise review of real world prediction problems and *Machine learning* tasks that can be extracted from them, we will briefly introduce supervised learning. In real world context, unlabeled data are more easily accessible than labeled examples. While supervised algorithms can only learn from labeled examples, semi-supervised algorithms reason from both labeled and unlabeled examples to learn a model. We will consequently discuss semi-supervised learning, along with some semi-supervised algorithms.

Machine learning algorithms are highly dependent on the relevance of the used metric and the representation space in which the data lies. On one hand, *representation learning* approaches propose to tackle the introduced problem by learning a new vectorial representation of the data. On the other hand, considering a set vectorial representation, the pairwise relationships between instances can be learned through *metric learning*. We will consequently finally discuss *metric* and *representation learning*, along with a brief review of *representation* and *metric learning* algorithms.

2.1 From real world problems to Machine learning tasks

A common task in real world problems is to be able to associate a continuous real value with an instance. Let us consider for example a commercial website budget forecast. Predicting the customers expenses for next year would probably entail the analysis of every customer purchase record from previous years and the similarities between customers. Attempting to do that manually would be very time consuming. Automatically predicting the

expenses of a customer based on his profile and purchase records would thus be a great improvement.

Another similar task that often needs to be solved in a real world context is to characterize some data with a predefined label. A nominal numeric label needs to be associated with an instance. Let us for example consider a commercial website that aims at proposing to a customer a set of interesting items. A solution is to automatically define if an item is likely to interest a customer and to associate with the item a label depending on the adequacy of the item with the customer without needing an expert review on items and customer.

Another common problem that needs to be solved is to define a partition of a set of instances. For example, a commercial website could seek at automatically creating groups of users based on their behaviours or common descriptors for market segmentation.

The various real world prediction problems that need to be solved automatically can be grouped under different categories depending on the *Machine learning* tasks that can be extracted from them. Among the different tasks introduced in Chapter 1 that can be extracted from real world problems, let us cite:

- **Clustering**: in clustering, problems are about defining a partition either over a set of instances or over the space in which lie the instances.
- **Classification**: the aim is to learn a function for classifying a set of instances by assigning them a label from a predefined set.
- **Regression**: in regression context, the aim is to learn a function qualifying the data, i.e. instances are associated with a quantitative value.

In our previously introduced commercial website recommendation example, the objective is to define the interest of a customer for a set of items. The dataset can be composed of a set of items. Labels in the recommendation problem should reflect the interest, or not, of the customer for the different items and can be defined as the binary set $\{0, 1\}$. The described real world problem can thus be transposed as a classification task.

For the budget forecast example, the dataset would be composed of a set of customers. The goal of the task is to associate a real value — amount of money — with each customer. The label set for the expenses prediction problem is the subset of positive real numbers $\{a \in \mathbb{R} | a \geq 0\}$. The customers expenses prediction problem can hence be considered as a regression task.

The market segmentation problem is to define a partition of the set of customers. The dataset being also defined as the set of customers, no a priori information is available for the market segmentation problem as there is no ground truth. The market segmentation problem can be seen as a clustering

task, where the set of customers aims at being partitioned.

In this work, we focus on classification task. An approach to solve a classification task is to take advantage of the available labeled examples to learn a classification function. Approaches reasoning from the set of available labeled examples lie in the domain of supervised classification. In supervised classification, by supposing a distribution $\mathcal{P}$ over the data representation space and the label set, a classification function g based on the information contained in the available labeled examples is aimed at being learned. The classification function g is learned to minimize the error on the available data used for training, ie. to satisfy at most the labeling on the available examples, and to minimize the generalization error, i.e. to generalize well to unseen instances.

Labels may originate directly from the real world problem context. In the commercial website budget forecast example, the expenses of the different customers during the previous years are indeed easily available from the website database.

More complex labels can be defined and such an information may not be directly available. In such cases, an expert can be asked to propose a review during or after the data collection process. Considering the problem of items recommendation for customers, the expert can analyse that a customer bought several books from an author. Based on the knowledge about the user the expert has extracted from the purchase records, the expert can indicate that the customer is likely to be interested for the latest book from the same author.

Real world data labeling can be expensive and non trivial. Due to labeling constraints, the amount of labeled data is often small compared to the whole set of available data. Real world dataset are often only partially labeled or not labeled at all.

Let us define our global notation for classification problems. Let $\mathcal{X}$ be a vectorial space and $\mathcal{Y}$ be a set of discrete labels $\{0, ..., c-1\}$. Let $\mathcal{P}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$. Let $\{(x_0, y_0), ..., (x_{l-1}, y_{l-1})\}$ be a set of labeled examples and $\{x_l, ..., x_{l+u-1}\}$ a set of unlabeled examples, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. We assume examples in both sets to be independent and identically distributed. Let us define $L = \{x_0, ...x_{l-1}\}$ and $U = \{x_l, ..., x_{l+u-1}\}$. Let us define the dataset X to be the union $X = L \cup U$. We assume examples in X were sampled according to $\mathcal{P}$. Let us finally define $\mathbf{y_L} = [y_0, ..., y_{l-1}]$ the labels vector of instances from L and $\mathbf{y_U} = [y_l, ..., y_{l+u-1}]$ the vector of label predicted for instances from U. The vector $\mathbf{y}$, which is the concatenation of $\mathbf{y_L}$ and $\mathbf{y_U}$ is then the labels vector associated with the dataset X.

Supervised classification consequently aims at learning a function g based

on $X = L$ ($U = \varnothing$) and $\mathbf{y_L}$. The classification function can then be used to predict the labels of unseen instances sampled from $\mathcal{P}$. It has however been highlighted that real world datasets can be only partially labeled, i.e. $X = L \cup U$. Supervised classification algorithms do not take advantage of the available unlabeled data to solve the task. With the growing amount of data, taking advantage of unlabeled instances could improve the task resolution.

Transductive learning approaches take advantage of both labeled examples and unlabeled instances to predict labels $\mathbf{y_U}$ of unlabeled instances. They however aim at solving the task only for unlabeled instances that are available at training time. Transductive learning does not seek at learning a general hypothesis on the instances domain and cannot predict any labels for unseen instances that were not available at training time. The classification function learned in transductive learning is consequently learned to minimize the error on unlabeled instances from U.

Semi-supervised classification algorithms also take advantage of unlabeled instances to solve the task. The learned semi-supervised classification function g does however not only focus on unlabeled instances available at training time as it aims at associating a label with any unseen instances from the distribution $\mathcal{P}$. Based on $X = L \cup U$, semi-supervised classification algorithms, similarly at supervised learning, aim at learning g to minimize the error on the examples used for training and to minimize the generalization error.

2.2 Supervised classification

A dataset X is composed of either a labeled set L, a unlabeled set U or both L and U. Labels of instances from L are either computed directly from the dataset creation process or based on an expert review. Based on a dataset X, inductive supervised and semi-supervised classification algorithms seek at learning a model for predicting a label for any instance drawn from the distribution $\mathcal{P}$. The targeted task can be solved for potential unlabeled instances U or unseen instances sampled from $\mathcal{P}$. This work focusing on semi-supervised classification, we only give a brief overview of the supervised classification domain

Supervised learning algorithms aim at retrieving the labels distribution of the dataset $X = L$ by producing hypothesis on the underlying probability distribution $\mathcal{P}$ of the instances. The produced hypothesis can be used to make prediction for unseen instances in $\mathcal{X}$ drawn from $\mathcal{P}$. Supervised learning classification algorithms learn a model mapping each instance of the domain $\mathcal{X}$ drawn from $\mathcal{P}$ to a label from the predefined label set $\mathcal{Y}$. Hypothesis are evaluated on unseen instances drawn from $\mathcal{P}$. More precise states of art of supervised learning can be found, notably in [Kotsiantis, 2007].

Many inductive supervised learning algorithms aim at learning the generative process explaining the labeled set L and are called **generative**. To cite some, *naïve Bayes* approaches ([Cestnik et al., 1987]), *mixture models* ([Dempster et al., 1977]), *restricted Boltzmann machines* ([Fischer and Igel, 2012]) or *linear discriminant analysis* ([Fisher, 1938, Fukunaga, 1990]) can be highlighted. Non generative algorithms are called **discriminative**. Some of discrimininative algorithms aim at learning a linear separator, among which *Support vector machines* ([Burges, 1998]) and *perceptron-based* algorithms ([Rosenblatt, 1962]) can be cited. Other algorithms, like kernelized version of linear separators, *neural networks* ([Zhang, 2000]), or *logic based* algorithms ([Murthy, 1998, Fürnkranz, 1999]) however aim at learning a non-linear separator for more complex datasets classification.

2.3 Semi-supervised classification

Supervised learning proposes various approaches in order to be able to predict a label for unseen instances sampled following the distribution $\mathcal{P}$, based on the available labeled data X. Learning process of supervised algorithms is exclusively based on the labeled instances that were available during the dataset creation. Due to real world constraints, including the labeling difficulty and cost, it has been highlighted that the proportion of labeled instances is often small considering the available data during the real world dataset creation. Supervised learning algorithms thus leave out most of the available instances during the training stage. This way, they do not enhance themselves with the information carried by unlabeled instances. Semi-supervised algorithms address the described issue by reasoning from datasets X composed of both labeled and unlabeled instances for the learning phase.

In order to efficiently exploit the unlabeled data U during the training, semi-supervised learning assumes that the underlying distribution of the data is somehow structured. Data is supposed to satisfy either the smoothness assumption or the cluster assumption ([Chapelle et al., 2006]).

In real world graphs, closely related entities are more likely to share common characteristics than distant ones. People are indeed more likely to be friends with people with similar tastes or opinions. The **homophily assumption** is the hypothesis that close instances are likely to be somehow similar. The real world based hypothesis of homophily was in particular discussed in [McPherson et al., 2001]. Based on the real world hypothesis, the *Machine learning* **smoothness assumption** is that two close instances are likely to be close considering the label, i.e. are likely to share a common label. Assuming as well that close instances are more likely to be similar in a way, the cluster assumption however supposes that data may tend to be spread in multiple

groups of approximately homogeneous labels.

Considering that at least one of the previous assumptions is satisfied, semi-supervised classification algorithms solve classification tasks by exploiting both labeled and unlabeled data. Semi-supervised learning is an important research field and many approaches have been proposed. In this work, we focus on a graph-based semi-supervised algorithm to solve the classification task. To place our work, let us briefly discuss semi-supervised classification approaches. Several surveys discuss about the numerous semi-supervised classification algorithms. Readers are referred to [Zhu, 2005, Chapelle et al., 2006] for more detailed discussion about semi-supervised learning.

Unlike supervised learning, semi-supervised learning takes advantage of the available unlabeled examples. There are various manners to exploit unlabeled data U to solve the task. On one hand, unlabeled instances, separated from the labeled ones, can be used as an informative bias for a supervised classification algorithm training. Unlabeled instances can on the other hand be considered as training instances and be used jointly with labeled examples during training stage.

Far from the algorithm we will focus on, a first part of semi-supervised classification algorithms use unlabeled data U of a dataset X to regulate the training step of a supervised classifier. Unlabeled instances can indeed exclusively be used to learn either a vectorial representation of the data, a kernel or a distance on the data. A supervised classifier can then be trained based on the newly learned component. Unlabeled instances can also be seen as potential future labeled examples. They can indeed be used to incrementally reinforce the labeled set used to train any supervised classifier or to evaluate the agreement between different learned models. Many **co-training** ([Blum and Mitchell, 1998]), **self-training** ([Yarowsky, 1995]) and **multi-view** algorithms ([de Sa, 1994]) hence exploit unlabeled data to influence the training phase of any supervised model.

Others semi-supervised classification algorithms jointly exploit labeled and unlabeled data to learn a model. Among them, **generative** algorithms, among which *expectation maximization* ([Dempster et al., 1977]), exploit the whole dataset, regardless of the existing labels, to estimate the conditional density underlying the dataset. Labels knowledge of instances from L are used to classify unlabeled instances sampled following $\mathcal{P}$, including instances from U.

Similarly to supervised learning, another group of approaches aims at learning a **discriminative** classifier. Discriminative algorithms aim at learning a decision boundary between the different classes based on labeled and unlabeled instances. Among the most known **discriminative** classification algorithms, let us cite *transductive support vector machines* ([Joachims, 1999]).

Part of semi-supervised algorithms reason from the relation between instances more than their vectorial representation. A graph representation of a dataset is a way to represent instances similarities relationships of a dataset as a set of instances and weighted edges. In such a representation, vectorial representations of the instances are ignored. A graph representation of a dataset can easily be built and many semi-supervised algorithms take advantage of that graph representation to solve the classification task. Algorithms taking advantage of the graph representation of a dataset are referred as **graph-based** methods. A more precise review of graph-based approaches will be given in Section 3.3.

2.4 Metric learning & representation learning

Problems like items recommendation can be expressed as a classification task where a label needs to be associated with a set of instances. A vectorial representation of the instances is defined on the real world problem context. The dataset is completed by associating labels with some or all the available instances. Numerous supervised and semi-supervised algorithms, assuming that the data satisfies either the smoothness or the cluster assumption, have been developed to solve the classification task associated with such a dataset.

Execution of classification algorithms are based at some point on the relationships between pairs of instances, which can be expressed through a distance their vectorial representations. Considering a dataset, algorithms are dependent either on the vectorial representation the data lies in and the distance associated with the vectorial space or on the pairwise relationships between instances. Pairwise relationships between two instances are commonly based on a distance between the vectorial representations of the instances. The distance associated with the representation space of the instance is supposed to be meaningful considering the task to solve. Distances are however generally chosen from a set of classical distances. Among them we can highlight the Euclidean distance which is the one most often used.

Depending on the targeted task, the relationship between two instances can vary. Classical distances were defined with the hypothesis that all features from the vectorial representation of an instance are equally relevant considering the relationship that aims at being expressed. The hypothesis of features equal influence cannot be ensured for all datasets and task combinations. Because of arbitrary choices during the dataset creation and vectorial representation creation, like the information selection or the numerical transposition for example, every feature is not ensured to have the same influence on the label than the others. As an example, several features of the representation may refer to the same initial information. For example, two features

extracted from the profile of a customer could be related to his gender: the gender — woman or man — and civility — M., Mrs. or Ms— features. We will refer to the presented phenomena as redundancy. Vectorial representation of a dataset can also be composed by features that are unrelated to the task, which will be called noise. Any information about one of the employees of the commercial website would for example probably be unrelated to the expenses of any customer and such an information in the vectorial representation of a customer will not help to solve the task. Due to noise and redundancy for example, an imbalance in the influence of the features on the label can be created, yielding to an inappropriate distance. There is thus no guarantees that the distance associated with the representation space — selected from the set of classical distances — is able to represent the similarity between the vectorial representations of two instances from the dataset for the task.

To overcome the highlighted shortcomings, many approaches have been developed in order to tune the vectorial representation of the data and the distance used to represent instances pairwise similarity. In the classification context, two instances are considered to be similar if their associated labels are identical. Adjusting the vectorial space of the dataset and the distance associated with it can be reached by either adapting the metric to the dataset representation (*metric learning*) or adapting the dataset representation to the metric (*representation learning*). In practice, the two approaches are closely related and *metric learning* is often fostered by a representation learning step. The algorithm that will be introduced in the Chapter 4 of this work aims at adapting the representation space of the data to a selected distance. Let us consequently briefly discuss about *metric learning* and *representation learning* algorithms.

To ensure that the used distance reflects the relationship required for the task, a first solution could be to set the vectorial representation of the dataset and to adapt the distance used to express instances similarity considering the targeted task. Some algorithms also neglect the vectorial representation by only considering the relationships between instances and the structure underlying the dataset. The objective is consequently to learn the most adapted distance, i.e. the pairwise relationship between instances, to reflect the similarity between instances, based on the fixed representation space. The introduced aim is in the scope of *metric learning*. Several algorithms have been proposed to learn a distance which is representative of the instances considering a specific task. Metric learning algorithms aim at learning pairwise relationships and consequently consider couples of instances. To learn the relationship between instances, they mainly reason based on a set of constraints that aims at being satisfied. A first group of guiding constraints are the must-link (respectively cannot-link) constraints that concern pairs of in-

stances. The must-link constraints (respectively cannot-link) reflect the fact that the selected pair of instances has to be considered as similar (respectively dissimilar) through the learned metric. Another kind of constraints are triplets relative constraints. A triplet relative constraint, commonly refered through the triplet (x_i, x_j, x_j), either simultaneously expresses a must-link constraint between x_i and x_j and a cannot-link constraint between x_i and x_k or expresses a partial ranking of the similarities between the pairs (x_i, x_j) and (x_i, x_k), i.e. that x_i has to be closer from x_j than from x_k. The set of constraints used to learn a metric can either be computed based the domain knowledge, on an expert review, or based on the task related supervision. Different detailed surveys discussing about *metric learning* have been proposed ([Yang, 2006, Kulis, 2012, Bellet et al., 2013]). Readers are referred to the introduced surveys for more detailed works about *metric learning* algorithms. Let us briefly discuss some *metric learning* algorithms.

Metrics can be learned either on the whole domain the dataset belongs to or on local patches. They are thus said to be either global or local. Global metrics may not be appropriate to represent heterogeneous data. Some algorithms were thus proposed in order to simultaneously learn multiple local metrics across the representation space of the data ([Frome et al., 2007, Ramanan and Baker, 2011]). A local metric can be learned either for each instance x_i from the training dataset or on local patches of the representation space of the data which can be defined based on the dataset.

All the datasets are not heterogeneous and learning multiple local metrics can be expensive for big scale datasets. Many metric learning algorithms rather seek at learning a global metric over the representation space of the dataset. Global metric learning algorithms can either learn a linear or a non-linear metric based on the previously introduced constraints.

Most commonly studied, numerous linear metric learning algorithms have been developed. Linear metric learning is however mainly about learning the matrix M parametrizing the Mahalanobis distance $d_M(x_i, x_j)$ between two instances x_i and $x_j \in X$ such that $d_M(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)}$. Numerous approaches aiming at learning a Mahalanobis distance or a variant have been developed. A well known algorithm for Mahalanobis distance learning is the one introduced in [Weinberger and Saul, 2009]. In this work, authors introduce the LMNN algorithm which learns a Mahalanobis distance based on two different sets of constraints $\mathcal{S}$ and $\mathcal{R}$. The set $\mathcal{S}$, similar to a set of must-link constraints, is composed of pairs (x_i, x_j) of instances x_i and $x_j \in X$ such that x_j is in the k-neighborhood of x_i and $\mathcal{R}$, which can be seen as a set of relative constraints, is composed of triplets (x_i, x_j, x_k) such that $(x_i, x_j) \in \mathcal{S}$ and that the labels y_i and y_k associated with x_i and x_k components of the triplet are different. The algorithm thus seeks at learning a Mahalanobis distance d_M that minimizes the distance between each pairs

$(x_i, x_j) \in \mathcal{S}$ while maximizing the margin between the pair of distances $d_M(x_i, x_k)$ and $d_M(x_i, x_j)$ of any triplet $(x_i, x_j, x_k) \in \mathcal{R}$.

Although linear metrics are convenient to optimize, they are not able to capture the non-linear structure of the data. Some approaches have been developed to learn non-linear metric. A first approach that has been applied for non-linear metric learning is the kernelization of linear metric learning algorithms. A linear metric learning algorithm is applied in a non-linear representation space induced by a kernel function ([Schölkopf et al., 1998, Scholkopf and Smola, 2001]). Such approaches are dependent on the choice of the kernel. Another group of approaches aims at directly learning a non-linear metric of the form $d_\phi(x_i, x_j) = d(\phi(x_i), \phi(x_j))$, where ϕ is a learned transformation function. The non-linearity is introduced by either learning a non-linear transformation ϕ with d being any selected classical distance or learning a linear transformation ϕ with d being a set non-linear distance. Let us for example cite the work introduced in [Kedem et al., 2012], where authors proposed two non-linear extensions of the previously introduced *LMNN* algorithm. The first approach they propose is to optimize the *LMNN* objective function for a non-linear distance. They secondly propose an approach seeking at learning a non-linear transformation of the data on which will be learned the Mahalanobis distance.

In order to learn a non-linear transformation function ϕ, ϕ can also be defined to be a neural network. For example, authors in [Chopra et al., 2005], in the *LSMD* algorithm, propose an approach to learn a non-linear transformation of the dataset. The approach aims at learning a convolutional neural network ϕ such that $d_\phi(x_i, x_j)$ is maximized for any pair $(x_i, x_j) \in \mathcal{S}$ the set of similar instances and $d_\phi(x_i, x_k)$ is minimized for any pair $(x_i, x_k) \in \mathcal{D}$ the set of dissimilar instances.

Defined as metric learning algorithms, previous approaches, in particular non-linear ones, can also be seen as mapping the data from the initial vectorial space to a representation space in which a distance satisfies some constraints. They are hence related to *representation learning*. Instead of trying to adapt the distance to the representation space, *representation learning* algorithms aim at directly mapping the dataset to a vectorial space in which the set distance is representative of the task similarity. *Representation learning* algorithms can be grouped depending on the set of features they aim at proposing.

First led by the objective to dispense with the expert review, *feature selection* algorithms propose to define subset of most relevant existing features of a dataset. The initial set of features may be irrelevant for the targeted task and a set of new features may need to be built by transforming the initial set of features. Among the *representation learning* algorithms aiming at building a set of more relevant features, *dimensionality reduction* algorithms constrain the number of created features to reduce the dimensionality of the

new vectorial representation, as many real world datasets have high dimensionality.

Our work being related to *representation learning*, let us briefly review some *representation learning* algorithms, in order to place our approach in a broader context. Several surveys detailing *representation learning* approaches are available ([Bengio et al., 2013, Guyon and Elisseeff, 2003]). Interested readers are referred to the two introduced surveys.

Based on the assumption that only a subset of features describing the instances are representative, reducing the dimensionality of our dataset or selecting the most representative features of the vectorial representation of our dataset could allow us to have a more adapted representation of our data. Such approaches are in the scope of *dimensionality reduction*.

Among the dimensionality reduction algorithms, *feature selection* approaches aim at keeping the most relevant variables describing the data, considering the targeted task. Part of *feature selection* algorithms aim at ranking the features describing the data by evaluating the individual influence of each feature on the labels. For example, authors in [Geurts et al., 2006], propose a classification algorithm growing a set of decision trees that can be used as a feature selection approach, which will be used in Section 6. A decision tree is a structure such that its several internal nodes correspond to a threshold, called the cut value, applied on an attribute. An unseen instance is classified by traversing the tree depending on the value of its features. The label associated with the instance is the label of the leaf the instance reaches. At each step of the tree learning, a set of attributes, and their associated cut values, are randomly sampled from the set of available attributes. The split performing the highest score, considering a fixed information gain measure, is defined as the new splitting node. Once the trees are growth, a score reflecting the importance of each feature depending on the information gain measure can be computed. A threshold can be applied on the computed scores for extracting the more relevant features of a dataset considering the specific task.

Some features may have a poor individual predictive power but may have a high predictive power when combined to some other features. As combining different distinct features can result to a set of features with a higher predictive power, several feature selection approaches aim at defining a subset of features have been developed.

The definition of a subset of relevant features can either be a preprocessing step unrelated to a given machine learning algorithm or be led by a given machine learning algorithm. When a specific machine learning algorithm is considered, the final subset of variables is selected either depending on its predicting power for the given algorithm — by evaluating the accuracy of solution — or its influence on the objective function optimized by the algorithm. More detailed works about *feature selection* can be found

in different surveys ([Guyon and Elisseeff, 2003]).

Other dimensionality reduction approaches, that can be grouped as *feature extraction* methods, aim at defining a smaller set of variables containing the same information than the input data for the task by transforming the initial features of the data. The features transformation can be either linear or non-linear. A subset of dimensionality reduction approaches are somehow related to the eigendecomposition of specific matrices computed based on the initial vectorial representation of the data. A well known spectral dimensionality reduction algorithm is the Principal Component Analysis (also known as *PCA*). The *PCA* algorithm aims at projecting the data in a new orthonormal basis and the new features describing the instances are the coordinate of the instances in the new system. The new basis is based on the eigenvector with the highest variance extracted from the matrix composed of the vectorial representation of the data. Neural networks are also used in dimensionality reduction. Well known neural networks for dimensionality reduction, are auto-encoders. An auto-encoder, and its associated auto-decoder, are networks that are learned simultaneously. An auto-encoder learns a smaller set of variables describing the data while the auto-decoder is learned to reconstruct the initial instance based on the representation learned by the auto-encoder, by minimizing the reconstruction error. Before solving a classification task, authors of [Rifai et al., 2011] propose to learn a vectorial representation of the data by learning an auto-encoder. They seek at learning an unsupervised representation which is robust to small input variations. For that purpose, a regularization term based on the sensitivity of the representation to the input, i.e. its derivative, is added to the usual objective function for auto-encoder.

Many other *representation learning* algorithms aim at learning a set of new features without dimensionality reduction constraints. Closely related to the approach that will be introduced in Chapter 4, remaining *representation learning* algorithms seeks at mapping our instances in a representation space by applying a transformation on the initial vectorial representation. Although some probabilistic approaches have been proposed, most of the representation algorithms without dimensionality reduction constraints handle neural networks to learn a vectorial representation of the data. By leaving out auto-encoders that were already discussed, neural networks based mapping functions can be learned based on similarity and dissimilarity constraints.

Among such approaches, let us for example cite the work proposed in [J. Weston, 2008]. The authors propose an algorithm to learn a multi-layered embedding based on a partially labeled dataset. They aim at learning the network by alternatively minimizing the loss between the embedding of two instances that are neighbours and maximizing the loss between the embed-

ding of a pair of instances where one is a labeled example and one is an unlabeled example. Work in [Hoffer and Ailon, 2014] can also be highlighted as the authors propose an algorithm seeking at learning an embedding of the data such that similarly labeled instances are pulled together when dissimilarly labeled instances are pushed away from each other. Both those works aim at learning a mapping neural network by minimizing a loss based on the embedding of different instances. They are consequently closely related to the algorithm that will be introduced in Chapter 4. The relationship and similarities between the different approaches are detailed in Section 4.5.

In the remainder of this work, we will discuss representation learning. A representation learning algorithm for graph-based semi-supervised classification algorithms will be introduced. The introduced representation learning algorithm will take advantage of neural networks to propose a representation of the dataset satisfying some metric related constraints.

Chapter 3

Graph-based semi-supervised learning

To solve the targeted *Machine learning* tasks, some of the semi-supervised algorithms neglect the vectorial representation of the dataset and reason from pairwise relationships between instances. Composed by a set of nodes and a set of weighted edges, a graph representation of a dataset is a representation of the dataset highlighting the pairwise relationships between instances. In a graph representation of a dataset, the vectorial representations of the instances are neglected. We will first discuss about graph characteristics. A general method to build a graph representation from a dataset is then introduced.

A subset of semi-supervised algorithms reasoning from pairwise relationships between instances directly learn from a graph representation of a dataset to solve the targeted task. The so-called graph-based algorithms assume that the graph they reason from at least satisfies the smoothness assumption, as introduced in Section 2.3. Properties graph-based algorithms expect from graphs and some graph-based semi-supervised algorithms are consequently reviewed. As we are going to see in the Section 3.2, given a dataset, several graph representations can be built. The smoothness assumption is not necessarily satisfied for the different available graph representations of a same dataset. We will finally discuss the influence of the construction methods on the execution of graph-based learning.

3.1 Graph representation

A graph is composed of a set of vertices that may be connected together by a set of edges. Let us formally define a graph:

Definition 1. *A graph G is a pair (V, E) where $V = \{v_0, .., v_{n-1}\}$ is the set of vertices and $E \subseteq V \times V$ is the set of edges linking the vertices.*

Let us call e_{ij} the edge in E linking the vertices v_i and $v_j \in V$.

Edges are used to express relationships between the vertices they are linking. They can express a set of different reciprocal relationships. An edge existing between two vertices can for example express either a symmetric similarity or dissimilarity between the two vertices. Edges can also be used to express a non reciprocal relationship. An edge from a vertex v_i to another vertex v_j can for example express the fact that the user represented by the first vertex v_i trusts the user represented by the second vertex v_j. The second user does not necessarily trust the first user and there can be no edge from v_j to v_i. Depending on the reciprocity of the edges, the graph is said to be directed or not.

For a set of nodes V, if the relationships reflected by the set of defined edges E are non-reciprocal, the associated graph representation $G = (V, E)$ is said to be directed and the edges are said to be oriented. In a directed graph $G = (V, E)$, if there is an edge e_{ij} starting from a vertex $v_i \in V$ to a vertex $v_j \in V$, the reciprocal edge e_{ji} starting from v_j to v_i may not belong to the set of edges E. On the opposite, the graph representation associated with a set of nodes V and a set of edges E reflecting reciprocal relationships between nodes is said to be undirected. In a undirected graph G, an edge starting from a $v_i \in V$ to $v_j \in V$ equates to an edge from v_j to v_i. For the remainder of this work, we will focus on undirected graphs.

Let us consider an undirected graph $G = (V, E)$. Let (v_i, v_j) and (v_k, v_l) be pairs of vertices from $V \times V$. Let e_{ij} and $e_{kl} \in E$ be the edges respectively linking each pair. A weight may be associated with every edges from E, allowing the relationships between instances to be qualified and/or quantified. In order to be somehow representative of the instances similarity, or dissimilarity, the weights associated with every edge are obtained by applying a function $f : V \times V \mapsto \mathbb{R}$ on the pair of instances being characterized.

Definition 2. *A weighted graph G is a graph $G = (V, E)$ where a real number w_{ij} is associated with each edge $e_{ij} \in E$.*

The real number $w_{ij} = f(v_i, v_j)$ is called the weight of the edge $e_{ij} \in E$.

Based on the introduced definition, a non-weighted graph is basically a graph for which the weights are identical for all edges. Non-identically weighted edges are able to express more complex relationships between instances by differently qualifying the relationship expressed by the edge associated with each pair of instances. For the rest of this work, let us consider all graphs to be weighted graphs.

Several matrix representations of a graph $G = (V, E)$ can be defined. Among the matrices that can be computed based on E and the weighting function f associated with G, let us consider the adjacency matrix representation of G. The adjacency matrix is a square matrix which rows and columns are indexed by the set of vertices V. The set of edges E linking

pairs of vertices from V can then be represented by weighting the entry associated with every pair of linked vertices of the graph. An entry on the $i-th$ row and the $j-th$ column in the adjacency matrix thus represents the weight of the edge linking the $i-th$ and $j-th$ vertices.

Definition 3. *Let $G = (V, E)$ be a graph such that $|V| = n$.*
The adjacency matrix $W \in \mathbb{R}^{n \times n}$ is the matrix such that

$$\forall i, j \in \{0, ..n-1\}, W_{ij} = \begin{cases} w_{ij} & \text{if } e_{ij} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

In an undirected graph $G = (V, E)$, if two vertices v_i and $v_j \in V$ are linked then both edges e_{ij} and e_{ji} belongs to E and $w_{ij} = w_{ji}$. The adjacency matrix of an undirected graph is a symmetric matrix.

3.2 Graph construction methods

Some real world data are already structured as graphs with explicit links between instances like for example datasets extracted from social networks. In such cases, it can be said that the graph representation is provided by the real world context. A lot of datasets however lack explicit graph representations.

In order to solve a task for a dataset through graph-based semi-supervised algorithms, an undirected graph representation of the dataset can be built. In the following, a classical methodology that can be applied to build an undirected graph representation of a dataset from its vectorial representation is provided in details. Different approaches to highlight the structure are then being discussed. The available results studying the introduced construction methods are discussed in Section 3.4.

A graph is by definition composed by a set of instances which are, at least, partially linked together by a set of edges. Let us consider a dataset X. To compute the graph representation $G = (V, E)$ of X, the set of vertices V and the set of edges E linking the vertices from V need to be defined. As G is the graph representation of X, the set of vertices V can be trivially defined as the set of instances X.

Graph representation of a dataset is built to reflect the pairwise relationships between the instances. The set of edges composing the graph representation associated with a dataset is constituted at most by the set of edges linking all the possible pairs of instances. The relationship expresses by an edge can be qualified by associating a weight with the edge. Based on a dataset, the most commonly available information about instances are their vectorial representations. The relationship between two instances can

be quantified, i.e. the edge linking two instances can be weighted, by applying a weighting function on the pair of their vectorial representations. Graphs being most of the time built to reflect relationships between the instances, the weighting function is commonly selected from the set of classical pairwise distance or similarity measures, like the Euclidean distance. A common function used as weighting function for the graph construction is the **gaussian radial basis function kernel**, based on the Euclidean distance, $f(x_i, x_j) = \exp\left(\frac{-d(x_i,x_j)^2}{\sigma}\right)$. Among the property of the gaussian kernel, the obtained weighted adjancy matrix is a similarity matrix, which is more likely to be handled by graph-based algorithms, as we will see in Section 3.3.

The previoulsy introduced gaussian kernel weighting function also displays a pruning advantage. Depending on the parameter σ, the value of the gaussian kernel for close instances, i.e. with a small pairwise distance, will be high while the weight associated with distant instances, i.e. with a high pairwise distance, will tend to 0. This way, negligible edges, i.e. relationships, are removed. The defined weighting function simultaneously performs a pruning of the built graph, by minimizing the importance, or removing, edges linking distant instances.

Let us now consider a distance f used to weight the graph representation $G = (X, E)$ of a dataset to be such that it does not prune the graph. We are interested in the relationships expressed in the graph representation of the data. Some edges may be negligible considering their weights compared to the other edges. Relative proximity of pairs of instances can also be drowned by the outnumbering quantity of other edges. The structure highlighted in the graph representation of the data should be emphasized. To stress the structure of the data, the edges that are the more representative of the instances relationships or the strongest relationships must be kept. Part of the created edges need to be removed in order to keep the more informative edges of G.

Pruning the graph G allows us to remove irrelevant or negligible edges. The pruning phase is consequently as crucial to the graph creation process as the weighting stage. The criterion reflecting if an edge is relevant or not is however not unique and may depend on a local or on a global approach. There are several methods in order to remove edges.

The weights associated with the edges of a graph somehow characterize the similarity between instances. Depending on the range of the weighting function f, the information carried by some weights is negligible. For example, considering a distance $f : V \times V \mapsto [0, 1]$, all weights close to 1 express little variation of a high dissimilarity between the characterized instances. Smallest variations can be neglected. Depending on the distribution of the

weights of the graph, a threshold separating the relevant information from the negligible one can be defined.

An approach to remove the poorly informative edges is the **ϵ-simplification**. ϵ simplification is a global pruning method where edges are kept depending on a threshold ϵ. An edge $e_{ij} \in E$ with a weight w_{ij} higher than ϵ is indeed removed from the set E. The distance between neighbours in such a pruned graph are ensured to be bounded by the threshold. The edges that are kept are consequently meaningful considering the absolute closeness of linked instances in the graph. As the edge removing policy does not depend on the connectivity of each vertex, there is no explicit knowledge about the obtained graph connectivity. The ϵ simplification method can lead to isolation of some instances which are distant from the whole dataset. The resulting graph can be composed of several small connected components.

With the previous approach, edges e_{ij} were neglected depending on the absolute value of their weights and a threshold. A weight which is negligible in a specific neighbourhood can be relevant considering the neighbourhood of another instance. A neglected edge e_{ij} can however be informative when compared to the set of edges associated with an instance x_i. Edges relevance can be analysed from a local approach.

Let us consider a graph $G = (X, E)$. The k-neighbourhood of each instance $x_i \in X$ can be defined as the set of k closest vertices that are linked to x_i by edges from E in G. The **k-nn simplification** is a local approach which reasons from the k-neighborhood of each instance of the graph. The importance of an edge depends on the mutual closeness of instances. An edge e_{ij} is informative if the instance x_j is in the k-neighbourhood of x_i. On the opposite, an edge e_{ij} is removed if x_j is not in the k-neighbourhood of x_i. For illustration purposes, let us consider the simple following graph G (Figure 3.1).

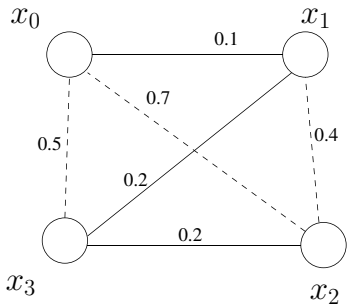


Figure 3.1 – Graph representation of a toy dataset

Let $X = \{x_0, x_1, x_2, x_3\}$ be a dataset and $G = (X, E)$ (Figure 3.1) be the undirected graph representation of X with a distance as weighting function.

Let $knn : X, G, k \mapsto X^k$ be the function associating with an instance x_i the set of k instances composing its k-neighbourhood in G. Let us consider $k = 2$ and let us compute the 2-neighbourhood of each instance:

$$\begin{cases} knn(x_0, G, 2) = \{x_1, x_3\} \\ knn(x_1, G, 2) = \{x_0, x_3\} \\ knn(x_2, G, 2) = \{x_1, x_3\} \\ knn(x_3, G, 2) = \{x_1, x_2\} \end{cases}$$

Applying the k-nn simplification of the graph G leads to a non-symmetric case where the edge e_{03} should be kept because $x_3 \in knn(x_0)$ but should be removed because $x_0 \notin knn(x_3)$. A similar non-symmetry appears considering x_2.

The graph representation obtained with the k-nn simplification in the previous example is a directed graph. To obtain an undirected graph, which are most commonly handled by graph-based algorithms, let us introduce two variants of the k-nn simplification leading to undirected pruned graph representations of a dataset.

The first variant is the **mutual k-nn simplification**. In the mutual variant of the k-nn simplification, an edge $e_{ij} \in E$ is kept if and only if the relation is meaningful for both instances, i.e. if and only if both x_i and x_j mutually belong to the k-neighbourhood of the other instance. In the toy example introduced in Figure 3.1, the resulting set of edges is defined as $E = \{e_{01}, e_{13}, e_{2,3}\}$. The introduced pruning method only keeps the edges that are the more meaningful considering pair of instances. Even if there is no explicit knowledge about the weight of the kept edges and their absolute importance, connectivity of each instance is ensured to be at most k. The minimal connectivity of the graph can not be ensured as the considered pruning method can lead to disconnected instances as both concerned instances have to be in the set of the closest instances from the other. Similarly to the ϵ-simplification, some instances can be disconnected and the obtained graph can be composed of several small connected components.

The second variant, the **symmetric k-nn simplification**, can be seen as a relaxation of the mutual one. With the symmetric k-nn simplification, an edge $e_{ij} \in E$ between instances x_i and $x_j \in V$ is indeed kept whenever the relation expressed by the edge is meaningful for at least one of the two concerned instances, i.e. if one of the instances is in the k-neighbourhood of the other instance. Considering our toy example, the resulting set of edges would be $E = \{e_{01}, e_{03}, e_{12}, e_{13}, e_{23}\}$. Although the weights of the edges are not explicitly constrained, symmetric k-nn simplification ensures that there is no isolated instance as every instance will be connected to at least k other instances. Let us define the floor function $\llcorner \cdot \lrcorner : \mathbb{R} \mapsto \mathbb{N}$ such that

$\forall z \in [a, a+1[, \llcorner z \lrcorner = a$. With the symmetric k-nn simplification, the resulting graph is ensured to have at most $\llcorner \frac{|X|}{k} \lrcorner$ connected components.

Finally, the common graph creation process where the pruning step is processed after the weighting function is summarized as in Algorithm 1. The described graph creation method is a general proposition, which iteratively performs each step.

Algorithm 1: Graph creation process

Data: the set of instances X
a weighting function f
a pruning method $simplify$
Result: $G = (V, E)$, W
`// Initialization`
$V = X$;
$E = \{\}$;
$W = \mathbb{R}^{|V| \times |V|}$;
`// Edges creation, weighting and pruning`
for $\forall x_i, x_j \in X$ **do**
  $E = E \cup e_{ij}$;
  $W_{ij} \leftarrow f(x_i, x_j)$;
end
$E \leftarrow simplify(V, E, W)$;
$G = (V, E)$, W;

The graph creation process summarized in Algorithm 1 performs the graph simplification once the whole set of possible edges is created and weighted. The described graph creation process is however not necessarily optimal, considering the memory and time needed. Depending on the pruning methods, the memory and time cost can be minimized.

An iterative process is mandatory for the mutual k-nn simplification, as all the edges related to instances x_i and x_j are needed in order to decide wether the edge e_{ij} is discarded or not. The mutual k-nn pruning method can only be applied once all the edges have been created and weighted.

On the opposite, ϵ-simplification keeps and removes edges depending on their absolute weights and on a set threshold. The ϵ-simplification can consequently be processed simultaneously to the edges creation and weighting step. In the symmetric k-nn simplification, the whole set of edges is not needed in order to prune the neighbourhood of an instance x_i. The symmetric k-nn pruning step can consequently be processed during the edges creation step. Such an iteratively created matrix needs to be symmetrized once all the instances neighbourhood have been processed.

Performing one of the two last pruning, ϵ or symmetric k-nn simplifica-

tion, jointly to the edges creation and weighting step allows us to minimize our memory needs. In those cases, the graph creation process is summarized differently.

3.3 Semi-supervised graph-based algorithms

A graph representation G of a dataset X can be built by choosing a weighting function f and a pruning method $simplify$. Some of the semi-supervised algorithms take advantage of the available graph representation G to solve the task associated with X. Let us assume f to be set, and W to be the adjacency matrix obtained based on G and f. As we will see in the present section, graph-based algorithms handle similarity matrices. If f is a distance, i.e. if its value is small for two similar instances, let us consider that a transformation was applied to map it into a similarity value. To transform the distance into a similarity value, several decreasing function, like the gaussian kernel can be applied. To tackle the targeted task, graph-based algorithms assume that G satisfies the following properties: smoothness and sparseness.

The main hypothesis graph-based semi-supervised algorithms rely on is that the smoothness assumption is satisfied on the considered graph. As introduced in Section 2.3, the smoothness assumption assumes that two close instances are likely to be close considering their labels. In a graph, two vertices are considered to be close if they are highly connected, i.e. the weight of the edge linking them is small (when f is a distance). Assuming that a graph satisfies the smoothness assumption leads to the assumption that vertices that are highly connected in a graph are similar considering the task. In a graph context, smoothness assumption is thus often considered as the fact that the labels are homogeneous, i.e. smooth, according to the graph structure. A graph G is consequently said to satisfy the smoothness assumption if two close instances in the graph are similar considering the labeling function, i.e. if the labeling function is smooth over G.

Real world graphs are usually not fully connected. In general their adjacency matrix is sparse. A pruned graph is also not fully connected and its adjacency matrix can be sparse. Graph-based semi-supervised algorithms assume that the sparsity property is shared by most of the graphs. They consequently exploit the sparse structure underlying the dataset to solve the task.

Let us recall some notations for the remainder of the present chapter. A set of instances $\{(x_0, y_0), ..., (x_{l-1}, y_{l-1})\}$ where $x_i \in \mathbb{R}^d$ with their associated labels $y_i \in \mathcal{Y}$ is available. The targeted task being the classification, $\mathcal{Y} = \{0, ..., c-1\}$ is a set of predefined labels. Let $x_l, ...x_{l+u-1}$ be a set of

unlabeled instances. Let $G = (X, E)$ be the graph representation of the dataset $X = L \cup U$ where $L = \{x_0, ..x_{l-1}\}$ and $U = \{x_l, ..., x_{l+u-1}\}$. Let W be the weighted adjacency matrix associated with G. Let us assume W to be a similarity matrix, i.e. W_{ij} is high for two similar instances x_i and x_j and small for two dissimilar instances x_i and x_j. Let $\mathbf{y_L} = [y_0, ...y_{l-1}]$ be the labels vector of the labeled instances of L. The aim of graph-based classification task is to be able to define a vector of labels $\mathbf{y_U} = [y_l, ...y_{l+u-1}]$ which associates a label $y_i \in \mathcal{Y}$ with each instance $x_i \in U$. Based on $\mathbf{y_L}$ and $\mathbf{y_U}$, let us define the vector $\mathbf{y} \in \mathcal{Y}^{l+u}$ the vector which associates a label of $\mathcal{Y}$ with each instance from X.
Let us introduce, in a binary classification context[1], the quadratic function

$$H(\mathbf{y}) = \sum_{i,j} W_{ij}(\mathbf{y}_i - \mathbf{y}_j)^2$$

The introduced function reflects how smooth the labels vector $\mathbf{y}$ is along the graph structure encoded in W. The quadratic function H for a labeling vector $\mathbf{y}$ where close instances in the graph have similar labels will tend to 0. A null value of the quadratic function for $\mathbf{y}$ indicates that every neighbouring instances have the same label so the labels vector $\mathbf{y}$ is optimally smooth along G. On the opposite, a high value reflects the fact that close instances have dissimilar labels depending on the labeling vector $\mathbf{y}$. Graphs are supposed to be homophilic structures. Labels along a graph are thus supposed to be smooth. The quadratic function of the labels of a dataset depending on its graph representation is supposed to tend to 0.

Led by the smoothness assumption, the objective of graph-based classification algorithms is to define a labeling such that it minimizes the interaction between dissimilarly labeled instances. Authors in [Blum and Chawla, 2001] propose a graph-based algorithm for binary classification, i.e. $\mathcal{Y} = \{0, 1\}$. Their aim is to define a partition of the set of instances X, based on a graph representation of X. Labeling is performed depending on the component instances belong to. They propose a specific graph construction in order to ideally partition the graph. Authors define $G = (X, E)$ to be the complete graph built from X. There are multiple ways to define the weighting function that is used to compute the graph. They discuss the influence of the chosen weighting function on the expected error. Based on the dataset $X = L \cup U$, they defined L_+ (respectively L_-) to be the set of labeled instances $x_i \in L$ such that $y_1 = 1$ (respectively $y_i = 0$). Two artificial nodes v_+ and v_- are added to the graph G. Each labeled instance $x_i \in L$ is linked to one of the two artificial vertices depending on its associated class, i.e. all instances

[1]It can be generalized to multi-classes problems as $H(\mathbf{y}) = \sum_{i,j} W_{ij}\mathbb{I}[\mathbf{y}_i \neq \mathbf{y}_j]$ where $\mathbb{I}$ is the indicator function, i.e. $\mathbb{I}[true] = 1$.

from L_+ are linked to v_+ and L_- are linked to v_-. The lastly created edges are added to E and weighted with an infinite weight.

A (x_i, x_j) cut of a graph G is a partition of the vertices of G such that the nodes x_i and x_j are in different subsets. By computing a minimum (v_+, v_-) cut on G through the max-flow algorithm, the authors can define a set of edges which disconnects v_+ and v_- if removed. By doing so, a two non-empty sets partition of the graph is obtained. The two sets contain either v_+ or v_- and are respectively called V_+ or V_- depending on the membership of either v_+ or v_-. Labels assignation is simply done by associating the class 0 with each instance contained in V_- and class 1 with each instance in V_+.

Even if the work presented in [Blum and Chawla, 2001] is not explicitly about optimizing the quadratic function, the function is still somehow minimized as each unlabeled instance x_i is assigned to the class set which is the most connected to x_i. The proposed approach indeed labels instances such that interactions between instances of the two different classes are minimized. The quadratic function of the initial G is thus minimized by the algorithm introduced by the authors.

Some approaches however tackle the classification problem through the direct optimization of the quadratic function H. Graphs are supposed to satisfy the smoothness assumption, considering the labels associated with the instances composing the graph. The unknown labels of unlabeled instances in the graph are also supposed to be smooth considering the graph and the labeled instances. The quadratic function H associated with the graph representation of a dataset is consequently supposed to be small. Graph-based classification algorithms hence propose a labeling $\mathbf{y}$ which minimizes the quadratic function $H(\mathbf{y})$ of the graph based on a matrix representation W of the graph and the known labels $\mathbf{y_L}$

The quadratic function H can be seen as the sum of the labeling dissimilarity between neighbouring instances, weighted by how close the pairs of instances are in the graph. An approach to directly minimize the quadratic function is to define labels for each unlabeled instance such that its neighbourhood disagreement on labels is minimized. The label propagation algorithm introduced in [Zhu and Ghahramani, 2002] computes a labeling of unlabeled instances by smoothing the labeling difference between neighbours instances. In their work, authors describe a method to iteratively minimize the quadratic function H of the graph representation of the input data. The idea of the algorithm is more precisely to associate with unlabeled instances of the graph a label which is based on the instance neighbourhoods. Labeling is done by propagating the available labels $y_i \in \mathbf{y_L}$ over the graph until labels for instances from U are stabilized. The final computed $\mathbf{y_U}$, which globally minimizes the quadratic function of the graph, thus tends to satisfy at best the smoothness assumption of the graph.

In the presented paper, considering the set of all the possible edges, authors assume that the weight w_{ij} of an edge e_{ij} is the gaussian kernel of the Euclidean distance between instances x_i and x_j. The adjacency matrix W associated with the graph representation of the data is a similarity matrix. Based on W, the authors compute a probabilistic transition matrix Q which is obtained by column-normalizing W, i.e. $Q_{ij} = W_{ij} / \sum_k W_{kj}$. The matrix Q is then itself row-normalized. Assuming a c classes classification problem, authors also introduce a $(l + u) \times c$ labels probabilities matrix F^0 such that for each labeled instance $x_i \in L$,

$$F_{ij}^0 = \begin{cases} 1 & \text{if } Y_i = j \\ 0 & \text{otherwise} \end{cases}$$

Initialization of F^0 for unlabeled instances is not important as long as $\forall i \in \{0, l + u - 1\}$, $\sum_k F_{ik}^0 = 1$. At each iteration, the matrix F^t is computed by propagating the labels probabilities distributions of the instances F^{t-1} along the matrix Q by computing the dot product $F^t = QF^{t-1}$. The probability of an unlabeled instance to belong to each class, at each iteration, is the mean of its neighbours probabilities distributions. Not to let the known labels $\mathbf{y_L}$ for labeled instances fade, the initial probabilities vectors of labeled instances are reset after each iteration. Let us define F^* to be the final probability matrix obtained. For any instance $x_i \in U$, the label associated with it is the class k with the highest probability F_{ik}^*. The several steps are summarized in Algorithm 2.

The authors show that, when t tends to infinity, the probabilities matrix F^t converges to a fixed limit which only depends on the known labels and the matrix Q. Let us define F_L^t to be the $l \times c$ matrix composed of the l first row of F^t which is the labels probabilities distributions matrix of labeled instances at iteration t. Let us similarly define F_U^t to be the labels probabilities distributions matrix of unlabeled instances. Q can similarly be decomposed as

$$\begin{bmatrix} Q_{LL} & Q_{LU} \\ Q_{UL} & Q_{UU} \end{bmatrix}$$

The authors show that the solution can be obtained with the following matrix formulation

$$F_U^* = (I - Q_{UU})^{-1} Q_{UL} F_L^0$$

Several variants of the work introduced in [Zhu and Ghahramani, 2002] have been developed. In [Zhu et al., 2003], restraining the task to a binary classification problem, the same authors extend their works to take advantage of external classifiers. They assume that an external classifier h is available

Algorithm 2: Pseudo-code of the label propagation algorithm

Data: $G = (X, E)$ the graph representation of the dataset $X = L \cup U$,
$\mathbf{y_L}$ the labels vector of labeled instances,
W the adjacency similarity matrix,
ϵ a convergence threshold
Result: $\mathbf{y}$ the predicted labels vector
`// Initialization`

$Q \leftarrow Q_{ij} = \frac{W_{ij}}{\sum_k W_{kj}}$

$Q \leftarrow Q_{ij} = \frac{Q_{ij}}{\sum_k Q_{ik}}$

$F^0 \leftarrow F^0_{ik} = \begin{cases} 1 & \text{if } x_i \in X \text{ and } y_i = k \\ 0 & \text{if } x_i \in X \text{ and } y_i \neq k \end{cases}$

Randomly initialise F^0 for $x_i \in U$ such that $\sum_k F^0_{ik} = 1$

`// Label propagation`

while $|F^t - F^{t-1}| > \epsilon$ **do**
  $F^t \leftarrow QF^{t-1}$;
  $\forall x_i \in L, \forall k \in \mathcal{Y}$, $F^t_{ik} \leftarrow F^0_{ik}$;
end
$\mathbf{y} \leftarrow \mathbf{y}_i = \max_k F^\infty_{ik}$

for unlabeled instances. To take advantage of the supplementary knowledge, they proposed to add an artificial node for each unlabeled instance $x_i \in U$ in the graph, carrying the label computed by the external classifier h for x_i. The edge linking each unlabeled instance to its associated artificial node is weighted by a selected parameter η. The hyper-parameter η expresses the importance given to the external classifier labeling. Let us define $\mathbf{h_U}$ to be the set of labels computed by h for the set of unlabeled instances. The label propagation algorithm can be applied on the augmented graph. The final solution is consequently of the form

$$F^*_U = (I - (1-\eta)Q_{UU})^{-1}[(1-\eta)Q_{UL}F_L + \eta\mathbf{h_U}]$$

Another variant of the label propagation algorithm aiming at computing $\mathbf{y_U}$ such that the labels $\mathbf{y}$ are smooth over the graph representation of the data, called *label spreading*, has been developed in [Zhou et al., 2004]. A first difference between the work from [Zhou et al., 2004] and the initially introduced label propagation algorithm lies in the definition of the matrix Q, which is defined such that $Q = D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$, where D is the diagonal matrix such that $D_{ii} = \sum_j W_{ij}$. The main difference however lies in the

operation computed during each iteration. In the introduced work, authors propose to update the probabilities matrix F^t depending both on the actual probabilities of each instance neighbourhood and on the initial probabilities of the labeled instances, by computing $F^t = \alpha Q F^{t-1} + (1-\alpha)F^0$. The parameter $\alpha \in [0,1]$ allows the authors to define the freedom the algorithm has to modify the labels of initially labeled instances. The last difference of the work from [Zhou et al., 2004] compared to the label propagation algorithm is that the introduced algorithm does not force the labels of labeled instances to stay at their initial values. The labels probability distributions of the initially labeled instances are prone to change during the several iterations. Initial labeling has a lower influence and the algorithm is more robust against error in the set of labeled instances. It should also be highlighted that the relaxation of the labeled instances labels allows more flexibility to minimize the quadratic function H of the graph. More flexibility on the labeling vector allows the graph to better satisfy the smoothness assumption, considering the labels of the instances. The authors then show that F^t converges to a fixed limit which can be written under the following form $F^* = (I - \alpha Q)^{-1} F^0$, which is pretty similar to the convergence limit of the label propagation algorithm.

The authors also propose a regularization framework for the described algorithm. They defined that the cost function $C(F)$ associated with their problem can be formulated as

$$C(F) = \frac{1}{2}\sum_{i,j=1}^{l+u} W_{ij} || \frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}} ||^2 + \mu \sum_{i=1}^{l+u} ||F_i - F_i^0||^2$$

where μ is a trade-off parameters to combine the smoothness related term and the fitting constraint. They show that $C(F)$ is minimized for

$$F = \frac{\mu}{1+\mu}(I - \frac{1}{1+\mu}Q)^{-1} F^0$$

Close from graph-based classification, a similar regularization framework has been defined for graph-based regression in [Belkin et al., 2004]. Authors define Q to be the Laplacian matrix of W at any power, i.e. $Q = (D - W)^p$ where $p \in \mathbb{N}$ and $D_{ii} = \sum_j W_{ij}$. Supposing that the labels $\mathbf{y_L}$ of instances from L can be noisy, the authors aim at solving the task with the Tikhonov regularization. More precisely, they aim at learning $F^* \in \mathbb{R}^{l+u}$ such that F^* minimizes the following cost

$$C(F) = \frac{1}{l}\sum_i (F_i - \mathbf{y_{L_i}})^2 + \gamma F^t Q F$$

where $\gamma \in \mathbb{R}$. Similarly to the previous paper, the cost is composed of a fitting constraint, which aims at retrieving the known labels, and of a

smoothness term which aims at minimizing the quadratic function H of the graph. Authors define $\mathbf{1} = [1, 1, ..., 1]$ the one vector and I the diagonal matrix of multiplicity of each instance, i.e. I_{ii} is the number of occurrences of the instance x_i in the labeled set. They show that the problem has a solution with

$$F^* = (l\gamma Q + I)^{-1}(\mathbf{y_L} + \mu\mathbf{1})$$

where μ is a parameter to be chosen such that F^* is orthogonal to $\mathbf{1}$.

Considering that the labels of $\mathbf{y_L}$ have no noise, the only goal to achieve is the smoothness assumption. Authors propose a second regularization framework to solve the task by minimizing the quadratic function. The solution vector F^* can thus be defined as

$$F^* = \underset{F=(y_0,y_1,..,y_{l-1},F^l,..,F^{l+u-1})}{argmin} F^T Q F$$

and can be obtained by

$$F^* = Q_{UU}^{-1} Q_{UL}^T ({\mathbf{y_L}}^T + \mu\mathbf{1})$$

All the previously introduced graph-based algorithms take advantage of the graph structure and the supposed smoothness assumption to solved the targeted task by somehow minimizing the variation of the labels along the edges. Based on an available graph representation of the dataset, some approaches take advantage of the graph in order to project the data in a representation space allowing the authors to perform a classical supervised or semi-supervised classification algorithm based on the vectorial representation of the data.

In [Perozzi et al., 2014], based on an input graph representation of the data, the authors propose an approach aiming at projecting the data in a representation space preserving neighbourhood properties of the graph. Based on random walks performed on the input graph, they aim at learning, for each instance in each random walk, the representation maximizing the probability of the instance to be in the specific random walk. They seek at taking advantage of the preserved smoothness assumption to perform a usual classification in the representation space.

A similar work is found in [Grover and Leskovec, 2016], where authors aim at learning data features such that the neighbourhoods of instances found in the network are preserved. Based on random walks over the graph to define a neighbourhood for each instance, the embedding of each instance are learned by minimizing a neighbourhood preserving objective function. A usual classification algorithm is then applied in the representation space of the data.

3.4 Graph construction relevance

Multiple graph representations of a dataset can be built to solve a real world problem, depending on some arbitrary choices that are made during the dataset or the graph creation processes. Considering a selected dataset for a real world problem, significantly different graphs can be obtained by choosing the vectorial representation of the data, the weighting function and the pruning method used for the graph construction step. Some of the built graphs are not ensured to satisfy the smoothness assumption considering the task. Such graphs are not meaningful considering the task to solve.

Graph-based algorithms take advantage of the proposed graph representation of the data and the supposed smoothness assumption to solve the task. The computation and the result of graph-based algorithms consequently depends on the relevance of the input graph. The arbitrary choices leading to the particular graph-representation of a dataset do have an influence on the task to be solved. Graph representation of a dataset is consequently a crucial point in graph-based semi-supervised learning.

In [Maier, Markus et al., 2013], authors make a theoretical study of the influence of graph construction methods on some graph-based algorithms computation. They more particularly work in a graph-based clustering context by analysing the behaviour of some clustering related quantities, like the normalized cut of a graph or some of its variants. For a partition of the graph $G = (X, E)$ in two sets of nodes A and $X \backslash A$, the cut value of the partition is the sum of the weights of the edges linking the instances from the different sets. The normalized cut, and its variants, are cost based on the cut value compared to the set of edges of the graph. The authors compare the convergence of the introduced measures on several graph representations of a same dataset. Clustering measures are compared for the k-nn simplification — regardless of the mutual or symmetric approach —, the ϵ simplification and the complete graph. Authors show that the differently built graphs lead to significantly different limits of the introduced cut quantities. The solution of a same task on different graphs built from the same dataset are consequently different.

Authors of [de Sousa et al., 2013] also work on the influence of graph construction for graph-based algorithms. They make an empirical study on the influence of the different arbitrary choices made during the graph construction. Accuracy of several graph-based semi-supervised learning algorithms are compared for different combinations of weighting functions and pruning methods. They also show that significantly different results are obtained from several graph representations of a same dataset.

Graph-based *Machine learning* algorithms rely on empirical graph construction methods for which few theoretical works have been performed.

Theoretical analysis of the influence of graph construction on algorithms execution are sparsely available. In the following of this work, a representation learning algorithm for graph-based classification is introduced. A theoretical analysis is performed to evaluate the interest of representation learning for graph-based classification. The work presented in this thesis take part to the objective of comprehending the influence of graph construction on task resolution.

In the next chapter, we introduce an algorithm to learn a more optimal representation space for graph-based classification algorithms. More precisely, the algorithm aims at learning a representation in which a targeted distance is representative of the task similarity. The graph built in the representation space learned with the the introduced algorithm with the targeted distance consequently should satisfy the smoothness assumption.

Chapter 4

A metric driven representation learning for graph-based label propagation

Any supervised and semi-supervised classification algorithm is somehow dependent on a distance applied on the vectorial representation of the data. Accuracy of supervised and semi-supervised algorithms is consequently dependent on the capability of the distance associated with the representation space of the data to reflect task similarity between instances. Can the vectorial representation of the data and the distance associated with it be tuned such that the distance on the representation space is meaningful considering the task to be solved?

Taking advantage of the relationships expressed by the graph representation, graph-based algorithms are a subset of semi-supervised algorithms ([Zhu, 2005]) that are particularly dependent on the adequacy of the distance to the vectorial representation of the data, as seen in Section 3.4. Graph-based semi-supervised algorithms, among which the algorithms introduced in Section 3.3, assume that the graph is a homophilic structure. As seen in Section 3.2, many graph representations can be built from a dataset. The graph representation of a dataset depends on choices made on the vectorial representation and the graph construction, notably the distance used to qualify the relationships between instances. Due to the numerous choices leading to a graph representation of a dataset, the smoothness assumption can hardly be ensured for any of the several graph representation that can be obtained for a dataset.

Trying to achieve the smoothness assumption on a graph equates to making each instance lies in a homogeneous neighbourhood considering the task. Smoothness assumption can be satisfied if the graph representation of a dataset is such that each instance is closer to similarly labeled instances than to

any dissimilarly labeled instance. In a graph, instances closeness depends on the distance used to weight the edges. The used distance should be representative of the task.

In the present chapter, we consequently aim at learning a distance d_ϕ on the instances such that it satisfies some relative constraints related to the smoothness assumption. In the following, we will first introduce a generic method to learn such a distance d_ϕ. For the distance d_ϕ to satisfy some constraints, the method we introduced aims at projecting the dataset in a representation space in which a predefined distance satisfied the set of constraints. More precisely, we will describe a method to learn a representation space of the dataset such that some relative constraints — related to the smoothness assumption — are satisfied. In a second step, we will describe how we implemented the generic approach by instantiating the representation that is learned. We choose artificial neural networks to learn a projection of the initial data. Artificial neural networks are increasingly used for representation learning. As graph-based classification requires to learn relationships between instances, a siamese architecture is necessary to learn our mapping function. We will consequently discuss about neural networks and the siamese architecture. The final goal is to solve a classification task through a graph-based algorithm. A graph representation of the data needs to be built in the representation space learned through the introduced algorithm. Let us consequently described the exploitation of the representation space for the graph-based classification. Several similar approaches have already been developed. Relationships of our approach to some similar algorithms are finally being discussed.

4.1 Representation learning for graph-based classification

The objective of the present chapter is to efficiently apply a graph-based classification algorithm on a graph representation of the dataset. The labels vector of the instances is consequently required to be smooth over the graph representation of the dataset. To build a graph representation of a dataset which satisfies the smoothness assumption, we aim at learning a distance d_ϕ on the instances such that similarly labeled instances are closer from each other than from dissimilarly labeled instances. More precisely, the distance d_ϕ is aimed at being learned such that $d_\phi(x_i, x_j) < d_\phi(x_i, x_k)$ for as many triplets of instances $(x_i, x_j, x_k) \in X \times X \times X$ such that $y_i = y_j \neq y_k$ as possible.

Learning a distance d_ϕ satisfying a set of constraints can be seen as learning a new vectorial representation ϕ of the instances such that a set distance d satisfies the introduced constraints in the representation space associated with ϕ. In this work, we consequently aim at learning a mapping function

ϕ projecting a dataset in a representation space such that a predefined distance d satisfies $d(\phi(x_i), \phi(x_j)) < d(\phi(x_i), \phi(x_k))$ for most of the triplets (x_i, x_j, x_k) such that $y_i = y_j \neq y_k$.

To learn such a mapping function ϕ, several optimization problems and loss functions can be defined. As the objective is to learn a distance between instances to build a graph, the loss function needed to learn the mapping function ϕ can be seen as a metric learning loss function. More precisely, the objective will be to learn ϕ to maximize the margin between $d(\phi(x_i), \phi(x_j))$ and $d(\phi(x_i), \phi(x_k))$ for as many triplets (x_i, x_j, x_k) such that $y_i = y_j \neq y_k$ as possible.

As artificial neural networks learn an internal representation of the data to solve prediction tasks, they are increasingly used to solve representation learning problems. Neural networks are also convenient due to their efficient optimization method called error-backpropagation. As the function ϕ that aims at being learned is a mapping function, let us focus on the subset of neural networks which can be seen as a direct mapping of the initial representation space.

Neural networks consider a single instance while we are interested in learning relationships between instances, i.e. we focus on tuples of instances. A siamese architecture provides a tool to handle multiple variables for a single neural network, allowing us to learn relations between instances by optimizing ϕ for a metric learning cost function. A siamese architecture is consequently developed in order to learn a neural network ϕ such that it minimizes the metric learning cost function.

The objective of this work is to outperform the initial vectorial representation of the data for a graph-based classification task. Once the mapping function ϕ is learned such that the data is projected in a representation space satisfying the more constraints as possible, a graph representation of the data is built in the presentation space mapped by ϕ, following one of the methods introduced in Section 3.2. A graph representation of the data being available, the classification task can be solved by applying the label propagation algorithm ([Zhu and Ghahramani, 2002]) described in Section 3.3.

The global approach we introduce for graph-based classification is summarized in Algorithm 3. We called it *MDRL* for *Metric Driven Representa-*

tion Learning.

Algorithm 3: MDRL: Representation learning algorithm for graph-based classification

Data: (L, y_L) a set of labeled examples and their associated labels
U a set of unlabeled instances
d a distance
Result: $\mathbf{y_U}$ a vector of predictions for instances from U

```
φ ← MDRL(L, y_L, d) // Representation learning
X̂ ← φ(X) // Projection of the dataset in the new representation space
G, W ← Graph Construction(X̂, d) // Graph representation construction in the new representation space
y_U ← Label Propagation(W, y_L) // Unlabeled instances classification
```

4.2 Constrained representation space learning

Before discussing our approach in more details, let us recall some notations. Let us recall $\mathcal{Y} = \{1, ..., c\}$ to be the set of predefined labels and $\mathcal{X} \in \mathbb{R}^p$ be a vectorial space. Let $(x_0, y_0), ..., (x_{n-1}, y_{n-1})$ be a set of labeled examples, where $\forall i \in \{0, ..., n-1\}$, $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. Let $X = \{x_0, ..., x_{n-1}\}$ be the set of instances and let $\mathbf{y} = [y_0, ..., y_{n-1}]$ be the labels vector of instances from X. Two instances x_i and x_j are said to belong to the same class if their labels y_i and y_j are equals. Let us finally introduce the Euclidean distance d_ϕ on $\mathbb{R}^p$.

To optimally perform a graph-based classification algorithm, the built graph representation of the dataset should be a homophilic structure. Our objective is consequently to learn the distance d_ϕ that allows us to build a graph representation of the dataset along which the instances labels are smooth. Labels are smooth considering a graph G if close instances in the graph are similarly labeled. We want to learn a distance d_ϕ such that for two instances x_i and x_j, if $d_\phi(x_i, x_j)$ is small (respectively large), the two instances are similarly (respectively dissimilarly) labeled. The distance d_ϕ used to build the graph representation of the dataset could be learned such that similarly labeled instances are closest from each other than from dissimilarly labeled instances. Considering $(x_i, y_i), (x_j, y_j), (x_k, y_k) \in X \times \mathcal{Y}$ such that $y_i = y_j \neq y_k$, d_ϕ should consequently be learned such that $d_\phi(x_i, x_j) < d_\phi(x_i, x_k)$.

Learning such a distance d_ϕ equates to learn a new vectorial representation $\phi(x_i) \in \mathbb{R}^q$ of instance $x_i \in X$ such that a defined distance d on $\mathbb{R}^q$ satisfies the smoothness related constraint for previously introduced instances x_i, x_j and x_k, i.e. $d(\phi(x_i), \phi(x_j)) < d(\phi(x_i), \phi(x_k))$. To learn

the new vectorial representation of the dataset, let us define ϕ to be a mapping function from the initial representation space $\mathbb{R}^p$ to a new representation space $\mathbb{R}^q$. Satisfaction of the relative constraint on the distances $d(\phi(x_i), \phi(x_j))$ and $d(\phi(x_i), \phi(x_k)$ can be achieved by learning ϕ such that the distance between dissimilarly labeled instances — $d(\phi(x_i), \phi(x_k))$ — is maximized and the distance between similarly labeled instances — $d(\phi(x_i), \phi(x_j))$ — is minimized. Maximizing an element while minimizing another one can be seen as maximizing the gap between the two elements. The mapping function ϕ can be learned such that, for the previously introduced instances $(x_i, y_i), (x_j, y_j)$ and (x_k, y_k), the difference between the two distances is positive, i.e. $d(\phi(x_i), \phi(x_k)) - d(\phi(x_i), \phi(x_j)) > 0$. Our goal can consequently be achieved by learning ϕ that maximizes the difference $d(\phi(x_i), \phi(x_k)) - d(\phi(x_i), \phi(x_j))$, i.e. margin between $d(\phi(x_i), \phi(x_k))$ and $d(\phi(x_i), \phi(x_j))$.

The dataset is however not only composed of the three introduced instances x_i, x_j and x_k. For the labels to be as smooth as possible over the graph, the objective is to satisfy the relative constraint on the distances $d(\phi(x_i), \phi(x_j)) < d(\phi(x_i), \phi(x_k))$ for as many triplets of labeled examples $(x_i, y_i), (x_j, y_j), (x_k, y_k) \in X \times \mathcal{Y}$ such that $y_i = y_j \neq y_k$ as possible. To fulfil as many relative constraints as possible for the dataset, the objective function to maximize could be

$$\sum_{(x_i,y_i),(x_j,y_j),(x_k,y_k) \in X \times \mathcal{Y} | y_i = y_j \neq y_k} d(\phi(x_i), \phi(x_k)) - d(\phi(x_i), \phi(x_j)) \qquad (4.1)$$

Maximization of the Equation 4.1 can allow us to maximize the distance of the negative pair and minimize the distance of the positive pair for each triplet that needs to satisfy the relative constraint. Such an optimization can however lead to inappropriate results. Maximization of such a sum can be achieved by highly maximizing some components of the sum and neglecting the other components. In our context, an extreme maximization of $d(\phi(x_i), \phi(x_k)) - d(\phi(x_i), \phi(x_j))$ can be reached for some triplets at the expense of other triplets. The difference between the distances for the neglected triplets could then be negative and the associated constraints would not be satisfied. Part of the relative constraints can be unsatisfied. The distance would be relevant only for part of the instances in the representation space. The graph would consequently not satisfy the smoothness assumption, likely to deteriorate the label propagation results.

To avoid a representation space satisfying only a small subset of the relative constraints, ϕ learning should be regulated not to encourage extreme cases. More precisely, ϕ should not be learned to radically maximize the difference $d(\phi(x_i), \phi(x_k)) - d(\phi(x_i), \phi(x_j))$ for any triplet of examples $(x_i, y_i), (x_j, y_j), (x_k, y_k) \in X \times \mathcal{Y}$ such that $y_i = y_j \neq y_k$. As the difference

between the negative and positive distances is only required to be positive, let us learn ϕ such that for any triplet of instances x_i, x_j and x_k as previously introduced, the difference between the negative and positive distances is bounded up to a selected margin $\mu \in \mathbb{R}$. Maximizing a difference up to a set margin equates to minimize the positive gap between the margin and the considered difference. In our context, for any triplet of examples $(x_i, y_i), (x_j, y_j), (x_k, y_k) \in X \times \mathcal{Y}$ such that $y_i = y_j \neq y_k$, we aim at minimizing the loss

$$\mu - [d(\phi(x_i), \phi(x_j) - d(\phi(x_i), \phi(x_k))] \tag{4.2}$$

A difference between the positive and the negative distances higher than the margin must be prevented. Not to exceed the defined margin, by minimizing the loss at extreme, let us consider the following hinge loss for any triplet of instances as previously defined:

$$max(0, \mu - [d(\phi(x_i), \phi(x_j) - d(\phi(x_i), \phi(x_k))]) \tag{4.3}$$

A hinge loss is a surrogate convex upper bound of the initial loss. The loss introduced in Equation 4.3 is clearly a hinge loss associated with the previous loss defined in Equation 4.2. The two losses are indeed equivalent for any instances x_i, x_j and x_j such that $d(\phi(x_i), \phi(x_j) - d(\phi(x_i), \phi(x_k)) \leq \mu$. For any instances x_i, x_j and x_j such that $d(\phi(x_i), \phi(x_j) - d(\phi(x_i), \phi(x_k)) > \mu$, the value of the loss from Equation 4.2 is negative, which is lower than the value of the loss from Equation 4.3. In the defined hinge loss, the difference between $d(\phi(x_i), \phi(x_k)$ and $d(\phi(x_i), \phi(x_j)$ can be maximized while it is lower than the margin μ. Whenever the margin is reached, the loss returns a zero score, indicating that the triplet does not need to be considered.

To satisfy the relative constraints associated with as many triplets of examples $(x_i, y_i), (x_j, y_j), (x_k, y_k) \in X \times \mathcal{Y}$ such that $y_i = y_j \neq y_k$ as possible, the loss from Equation 4.3 aims at being optimized for all the possible triplets of instances $(x_i, x_j, x_k) \in X$ satisfying the introduced equality constraint on their labels. The relative constraints on which is based the representation learning are computed on a set of triplets (x_i, x_j, x_k). A specific semantic is associated with the order of the instances described by a triplet. The first component of the triplet can be considered as a pivot point, the second component is more similar to the pivot than the last component. More precisely, in the representation space that aims at being learned, the first component x_i of a triplet (x_i, x_j, x_k) is expected to be closer from x_j than to x_k, considering the distance d.

Definition 4. *Let us define T to be a set of triplets of instances such that for all triplets $(x_i, x_j, x_k) \in T$, x_i is expected to be closer to x_j, in the new space, than to x_k:*

$$T = \{(x_i, x_j, x_k) | x_i, x_j, x_k \in X\}$$

The set of triplet T as defined in Definition 4 can be obtained from various methods. Such triplets can for example be gathered from the data collection process itself. For temporal videos, two successive images have high probability to be more similar than two non-successive images. Easily accessible triplets could be computed from a temporal video dataset. The set of triplets could also be based on domain and experts knowledge. Experts could indicate specifics similarity and dissimilarity rules.

In our context, the set of triplets can be based on the labels of the instances and our set T can be computed directly based on the task. Relative constraints that aims at being satisfied will consequently be directly based on the set of triplets composed of two instances from a same class and an instance from a different class. From a dataset X and the associated labels $\mathbf{y}$ as initially introduced, one can always compute the following set of triplets

$$T = \{(x_i, x_j, x_k) | x_i, x_j, x_k \in X \wedge y_i = y_j \neq y_k\} \tag{4.4}$$

To satisfy as many constraints as possible based on the whole set of triplets T, the loss from Equation 4.3 is optimized for all the triplets from the set T. The mapping function ϕ is consequently learned by minimizing the following cost function:

$$C(\phi|T) = \sum_{(x_i,x_j,x_k)\in T} max(0, \mu - [d(\phi(x_i), \phi(x_j) - d(\phi(x_i), \phi(x_k))]) \tag{4.5}$$

where μ is the margin.

Minimization of the sum of bounded distances differences between pairs of instances cannot lead to extreme maximization of some of the terms. The distance is consequently more homogeneously representative of the task similarity in the mapped representation space. To project our data in a representation space allowing us to build a graph satisfying the smoothness assumption, the mapping function ϕ is learned to minimize the introduced cost. More precisely, considering a dataset X and its associated set of triplets T, ϕ is learned such that

$$\phi = \underset{f:\mathbb{R}^p \mapsto \mathbb{R}^q}{\arg\min}\ C(f|T)$$

4.3 Non-linear neural network learning

The aim of the approach is to learn a new vectorial representation of the data. Artificial neural networks are increasingly used for representation learning due to their internal representation learning and their efficient optimization method. Some neural network architectures are more related to representation learning as they can be seen as a direct mapping function from the

initial vectorial representation to a new representation space. Let us first describe the subset of neural networks we are interested in and let us assume for the remainder of the work that ϕ belongs to the introduced subset.

Neural networks only consider one instance as input, while our approach aims at learning a relation between instances. To handle several instances, and more precisely triplets of instances, let us introduce the siamese architecture.

Multi-layers perceptrons

A neural network can be seen as a function mapping an input variable to an output variable. Among the set of existing neural networks, multi-layers perceptrons, or multi-layered perceptrons, are processing structures commonly used for supervised learning. They are composed by different successive layers, the first and the last layers being respectively named the input and the output layers. Intermediate layers are called the hidden layers. Also called feed-forward neural networks, multi-layers perceptrons constitute a subset of neural networks where each layer is connected to the previous and the next layers. There is consequently no cycle in such neural networks.

The layers of a neural network are themselves composed by a set of units, called neurons. A perceptron ([Rosenblatt, 1958]) is a unit which produces a binary output by applying a threshold on a weighted combination of its inputs. As a generalization of the perceptron, neuron (Figure 4.1) is a unit producing a continuous value by applying a (commonly) non-linear function, called activation function, on a weighted linear combination of its inputs. The activation functions that are commonly used are the sigmoid or the hyperbolic tangent. Successive layers of multi-layers perceptrons are

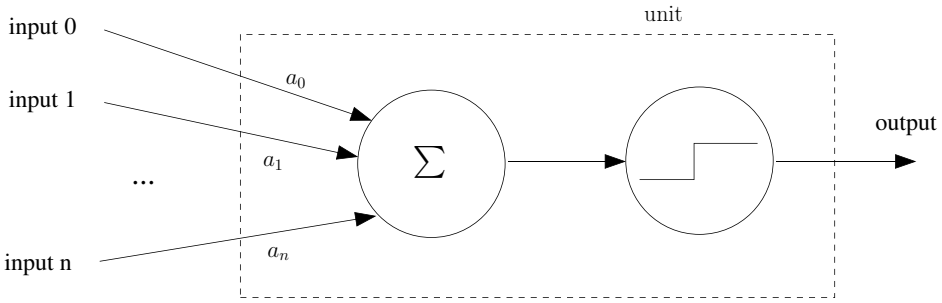


Figure 4.1 – Example of a neuron

connected through their units. The inputs of any neuron of such a multi-layers perceptron is the set — or a subset — of the outputs produced by the units from previous layer. Similarly, the output of any neuron is duplicated to be one of the inputs of each neuron — or a subset of neurons — from the next layer.

Let us consider a multi-layers perceptron composed of l successive layers, where each layer h is composed of k_h units. Layers 0 and $l-1$ are respectively the input and output layers. Let us define u_i^h the $i-th$ unit from layer h. Let A_i^h be the vector of weights associated with the unit u_i^h and let g_i^h be the activation function associated with each unit u_i^h. Let $[x_0, ..., x_{k_0}] \in \mathbb{R}^{k_0}$ be an input variable. Based on the introduced notation, the output of any unit i from the input layer 0 is $u_i^0([x_0, ..., x_{k_0}]) = g_i^0(x_i)$. More generally, the output of each unit i from layer h can consequently recursively be defined as

$$u_i^h([x_0, ..., x_{k_0}]) = g_i^h \left(\sum_{j=0}^{k_{(h-1)}-1} A_{ij}^h u_j^{h-1}([x_0, ..., x_{k_0}]) \right)$$

Figure 4.2 is an example of a fully connected multi-layers perceptron. The output of each unit can be expressed as a function of the initial inputs and the weights of the network. Such a neural network consequently corresponds to a non-linear projection from an initial representation space to a new representation space. Multi-layers perceptrons can reasonably be used as mapping functions from an initial representation space to a new vectorial representation.

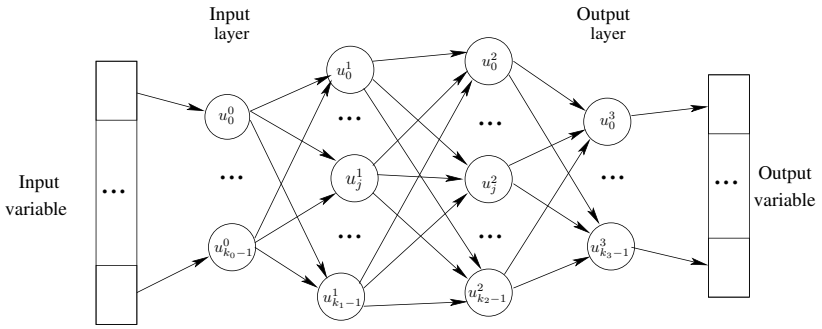


Figure 4.2 – Example of a fully connected multi-layers perceptron

Various theorems have stated about the powerful representational capabilities of non-linear feed-forward neural networks ([Hornik et al., 1989]). Let $A^h = [A_0^h, ..., A_{k_h-1}^h] \in \mathbb{R}^{k_{h-1} \times k_h}$ be the matrix composed of the vectors of weights of each unit from layer h. A multi-layers perceptron is parametrized by its depth and width hyperparameters — the number l of layers and the number of units k_h composing each layer h — and the set of weights matrices $\{A^1, ..., A^{l-1}\}$ associated with its several layers. For the rest of this work, let Φ be the set of multi-layers perceptrons with a fixed architecture. Learning the mapping function ϕ through our algorithm equates to learn the optimum set of weights matrices $\{A^1, ..., A^{l-1}\}$ for the fixed architecture.

Learning

A neural network ϕ associates a new variable $\phi(x)$ to an input variable x. Depending on the goal driving the neural network learning, a cost function $C(\phi(x)|x)$ can be defined to evaluate the relevance of the neural network output. The objective is to learn the network ϕ to minimize the value of the cost function. In addition to being able to handle complex datasets, a powerful and computationally efficient method for neural networks optimization has been defined.

We have seen that a multi-layers perceptron with l layers is parametrized by its set of weights $\{A^1, ..., A^{l-1}\}$. The influence of each of the weights on the cost function can be defined. To minimize the value of the cost function, the weights of the network can be updated depending on their impact on the obtained error. An optimization method to update the weights of the network is called the error backpropagation, mainly introduced in [Lecun, 1985, Rumelhart et al., 1986, Le Cun, 1986]. The error backpropagation optimization method is an iterative approach, where the weights of the network are updated depending on the error obtained considering the input instance. During the learning phase, each weight of the network is adjusted by subtracting a proportionate rate of its associated derivative depending on a learning rate $\alpha \in \mathbb{R}$, i.e. $A^h_{ij} = A^h_{ij} - \alpha\Delta^h_{ij}$. The learning rate is defined in order to adjust how fast the parameters are updated at each step. Δ^h_{ij} corresponds to the influence of each weights A^h_{ij} on the obtained cost and can be computed as

$$\Delta^h_{ij} = \frac{\partial C(\phi(x)|x)}{\partial A^h_{ij}}$$

The term $\frac{\partial C(\phi(x)|x)}{\partial A^h_{ij}}$ can itself be decomposed as

$$\begin{aligned}\frac{\partial C(\phi(x)|x)}{\partial A^h_{ij}} &= \frac{\partial C(\phi(x)|x)}{\partial \sum_{m=0}^{k_h} A^h_{mj} u^{h-1}_m(x)} \times \frac{\partial \sum_{m=0}^{k_h} A^h_{mj} u^{h-1}_m(x)}{\partial A^h_{ij}} \\ &= \frac{\partial C(\phi(x)|x)}{\partial \sum_{m=0}^{k_h} A^h_{mj} u^{h-1}_m(x)} \times u^{h-1}_i(x)\end{aligned}$$

Let us define, for a layer h, $\delta^h_j = \frac{\partial C(\phi(x)|x)}{\partial \sum_{m=0}^{k_h} A^h_{mj} u^{h-1}_m(x)}$. Δ^h_{ij} can consequently be expressed as

$$\Delta^h_{ij} = \delta^h_j u^{h-1}_i(x)$$

To define the influence Δ_{ij}^h of each weight A_{ij}^h from hidden layer h, the expression of the term δ_j^h must be defined for all the hidden layers. Each unit j from layer h influences the final error as an input for units from next layer. Each derivative δ_j^h for unit j from hidden layer h can consequently be recursively defined depending on the next layer. The term δ_i^{h-1} associated to the $i-th$ unit from layer $h-1$ partially depends on the influence of its successor units, among which the unit u_j^h. Based on the chain rule for partial derivative, considering a specific successor unit u_i^h:

$$\frac{\partial C(\phi(x)|x)}{\partial \sum_{m=0}^{k_h} A_{mj}^h u_m^{h-1}(x)} = \frac{\partial C(\phi(x)|x)}{\partial \sum_{m=0}^{k_h} A_{mi}^h u_m^{h-1}(x)} \times \frac{\partial \sum_{m=0}^{k_h} A_{mi}^h u_m^{h-1}(x)}{\partial u_i^{(h-1)}(x)} \times \frac{\partial u_i^{(h-1)}(x)}{\partial \sum_{m=0}^{k_{h-1}} A_{mj}^{h-1} u_m^{h-2}(x)} \tag{4.6}$$

To incorporate the influence of each unit u_i^h from the next layer, the quantity defined in Equation 4.6 is summed over all the units of the successor layer:

$$\delta_j^{(h-1)} = \sum_{i=0}^{k_h} \left(\frac{\partial C(\phi(x)|x)}{\partial \sum_{m=0}^{k_h} A_{mi}^h u_m^{h-1}(x)} \times \frac{\partial \sum_{m=0}^{k_h} A_{mi}^h u_m^{h-1}(x)}{\partial u_i^{(h-1)}(x)} \times \frac{\partial u_i^{(h-1)}(x)}{\partial \sum_{m=0}^{k_{h-1}} A_{mj}^{h-1} u_m^{h-2}(x)} \right)$$

which can be simplified as follows:

$$\delta_j^{h-1} = \frac{\partial u_i^{(h-1)}(x)}{\partial \sum_{m=0}^{k_{h-1}} A_{mj}^{h-1} u_m^{h-2}(x)} \sum_{j=1}^{k_h} \delta_j^h \times W_{ij}^h$$

Each weight A_{ij}^h parametrizing the multi-layers perceptron can consequently be updated by computing $A_{ij}^h = A_{ij}^h - \alpha \delta_j^h u_i^h(x)$.

In our setting, let us consider a dataset $X \subset \mathbb{R}^p$. The feed-forward neural network that aims at being optimized in order to map our data into a representation space $\mathbb{R}^q$ satisfying some relative constraints will be characterized by an input layer with p units, i.e. $k_0 = p$, and an output layer with q units, i.e. $k_{l-1} = q$. The depth of the neural networks, i.e. the number of hidden layers, and the width of each layers, i.e. the number of units, will be empirically chosen.

Siamese neural network architecture

In this work, we are interested in learning a relationship between instances, and more precisely a distance between the vectorial representations of two instances. To learn the appropriate mapping function, the cost function that aims at being minimized handles the projection of a triplets of instances. The introduced cost function handles the output provided by ϕ for three distinct input variables. To optimize such a loss, a specific neuronal architecture needs to be defined. In [Bromley et al., 1994], an architecture is introduced to simultaneously manage the application of a same network on distinct inputs. A siamese neural network is an architecture composed of as many similar sub-networks as distinct inputs. A comparator is added on top of the dupplicated networks to compare the output obtained from each of them in order to evaluate the final cost function. Figure 4.3 illustrates the architecture with a small example.

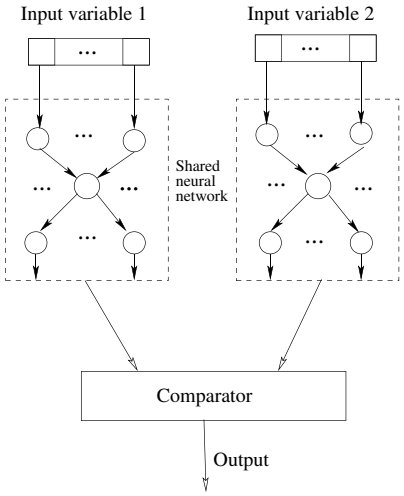


Figure 4.3 – Example of a siamese neural network

In our setting, our cost function is composed of a comparator, the hinge loss which compares two inputs a and b: $max(0, \mu-(a-b))$. The comparator layer composing our global network takes as inputs the pairwise distances $d(\phi(x_i), \phi(x_j)$ and $d(\phi(x_i), \phi(x_k))$. The compared distances correspond to the distance between two instances in the representation space mapped by ϕ. In order to compute the distance between two instances in the new representation space, two copies of the feed-forward network representing the mapping function ϕ are needed, complemented by a distance layer. The previously defined module can itself be seen as a siamese neural network with two inputs, x_i and $x_j \in \mathbb{R}^p$, and a unique output $d(\phi(x_i), \phi(x_j)$ (Figure 4.4).

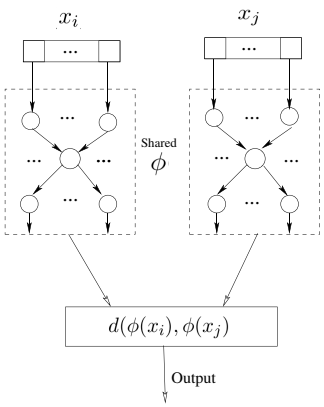


Figure 4.4 – Example of a siamese architecture for a pairwise distance.

The comparator layer computing the hinge loss takes as inputs the output of two copies of the previous module. The siamese network needed to compute the loss in performance associated with a triplet of instances $(x_i, x_j, x_k) \in T$ is thus composed of two copies of a smaller siamese network. The sub-network composing the siamese architecture is composed of a distance layer on top of two copies of the feed-forward network corresponding to the mapping function. The whole siamese network is consequently composed of four copies of the feed-forward neural network ϕ.

The instance x_i is shared by the two pairwise distances that are being compared in the final loss function. The inputs of our siamese architecture are consequently two pairs of instances, one being shared. The output of the described network is the loss function evaluated on a triplet $(x_i, x_j, x_k) \in T$, aiming at being minimized in order to learn ϕ. The global siamese network that is defined to learn the mapping function ϕ can be summarized in Figure 4.5.

Optimization of the mapping function ϕ by backpropagation in a siamese architecture.

The siamese architecture previously introduced allows us to evaluate the loss from Equation 4.3 for a triplet $(x_i, x_j, x_k) \in T$ considering the actual mapping function ϕ. To learn ϕ such that it minimizes $C(\phi|T)$, weights of ϕ must be modified depending on the error obtained for a specific input. In our setting, the input is constituted of three distinct instances. Each feed-forward neural network projecting one of the instances in the new representation space needs to be updated depending on its influence on the

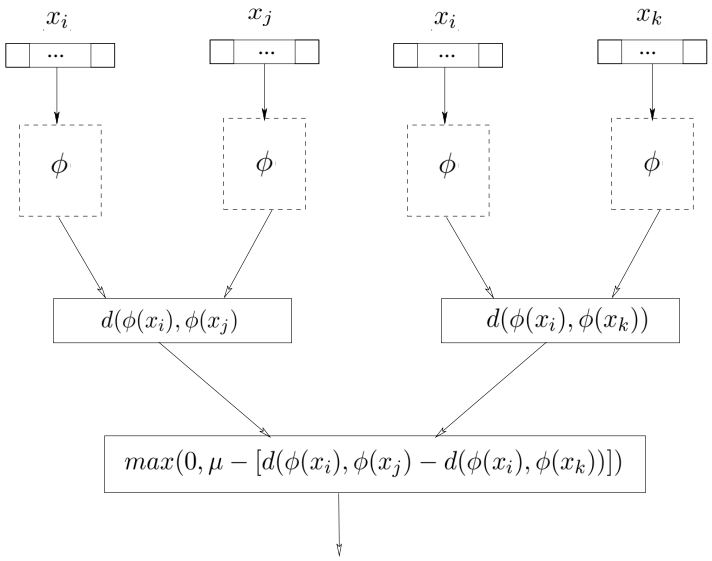


Figure 4.5 – Siamese neural network framework for a mapping function ϕ, a distance d, a margin μ and a triplet $(x_i, x_j, x_k) \in T$

error. All the several sub-network are however the same shared feed-forward network. To update ϕ depending on the impact of each instance of a triplet, the derivative of the error depending on each weight of ϕ is iteratively back-propagated for each instance of the triplet.

4.4 Graph-based classification

Stochastic gradient descent on the siamese neural network for the loss evaluated on triplets of instances from T allows us to update parameters to minimize the cost function C. The only tunable weights of the introduced siamese architecture are carried by the feed-forward neural network corresponding to our mapping function ϕ. The iterative backpropagation of the cost function evaluated on random triplets (x_i, x_j, x_k) from T allows us to obtain a mapping function ϕ that minimized our global cost function $C(\phi|T)$.

Considering the learned function ϕ, mapping a dataset $X = L \cup U$ from $\mathbb{R}^p$ to $\mathbb{R}^q$, and a label vector $\mathbf{y_L}$, let us described how classification of unlabeled instances from U is performed, i.e. how the vector $\mathbf{y_U}$ is computed.

This work focuses on graph-based classification algorithm and more particularly on the algorithm introduced in [Zhu and Ghahramani, 2002][1]. As seen in Section 3.3, the set algorithm iteratively propagates the label of each

[1] implemented in *Scikit-learn* (Python module, [Pedregosa et al., 2011])

instance to his neighbourhood until stabilisation. Neighbourhood is defined depending on the structure of the graph the algorithm is applied on. A graph representation of the dataset first needs to be built before being able to classify unlabeled instances from U.

As seen in Section 3.2, a common graph representation G construction method is to weight all the pairs of instances from the dataset X. Weighting is commonly done by applying a distance on the vectorial representation of the considered pair of instances. In our setting, the representation space of the data has been learned to outperform the initial representation. The graph representation is computed in the representation space the dataset is projected in. The mapping function ϕ has moreover been learned depending on a fixed distance, the Euclidean distance in our setting. The weighting function used to weight the graph representation of the dataset in the representation space mapped by ϕ is consequently the Euclidean distance.

In the same section, we have seen that several pruning methods can be applied to keep the most relevant edges of a graph. An intuition about our approach is that the representation space is learned such that an instance is closer, depending on the Euclidean distance, to any similarly labeled instance than to dissimilarly labeled instances. A threshold can be assumed such that two instances with a distance beyond the defined threshold have a high probability to be dissimilarly labeled. Let us consequently focus on the ϵ-simplification in our setting.

Once such a graph-representation of the dataset X is built, the label propagation algorithm can be applied to predict the labels vector $\mathbf{y_U}$ for instances from U. The process starting from the initial space of the data and leading to the classification of instances from U has already been summarized in previously introduced Algorithm 3.

4.5 Related work

Among the metric learning and representation learning algorithms that have been developed (cf Section 2.4), some of them are driven by a supervised or semi-supervised task. The targeted task supervision can be variable. In [Rifai et al., 2011], authors propose an unsupervised representation learning based on auto-encoder neural networks. After the auto-encoder training, a classification layer is added. The whole network, among which the representation layers, is fine-tuned before a supervised classifier training. The representation learning is poorly supervised by the targeted task. Authors of [J. Weston, 2008] also perform a representation learning in order to solve the targeted task. They attempt a direct learning of both the representation and

the classifier. The representation is learned to improve the model used for classification. Representation learning is either treated as an auxiliary task of a classifier training or is performed simultaneously to the classification training.

Some approaches, close to the algorithm that will be introduced, directly bias the learning phase with task related constraints. For example, authors in [Chopra et al., 2005] propose a task driven representation learning in order to solve a supervised classification. They project the data in a new representation space by learning a convolutional network. The weights of the networks are learned based on pairwise similarity and dissimilarity constraints defined based on the classification task. The classification task is solved by directly comparing the vectorial representation of two pictures. The pairwise constraints being computed directly from the task labeling, the supervision of the representation learning algorithm is directly related to the targeted task. Similarly to the described appproach, authors of [Hoffer and Ailon, 2014] also proposed a supervised representation learning for a classification task. More precisely, they train a neural network, through triplet constraints, to project the data in a new representation space. They finally train a one-layer network on top of the representation to solve the classification task. The representation is also learned with task related supervision.

Algorithms proposed in [Rifai et al., 2011] and [J. Weston, 2008] are, similarly to our work, learning a representation before solving the task. The representation is however fine-tuned during the supervised classifier training, while the representation learned by our *MDRL* algorithm is not modified during semi-supervised classification. Another difference of [Rifai et al., 2011] and [J. Weston, 2008] with our approach is that, in their works, the classifier is parametric while we rely on a non-parametric classifier on which we give guarantees. In [J. Weston, 2008], they proposed an approach attempting a task driven representation learning by learning both the representation and the classifier, which is pretty different from our work. [Chopra et al., 2005] and [Hoffer and Ailon, 2014] also learn the representation of the data before solving the supervised task. Work proposed in [Hoffer and Ailon, 2014] however differ from our work by also proposing a parametric classifier, on the opposite to our non-parametric classifier. Finally, the main difference of our approach with that of [Chopra et al., 2005], is the shape of their representation function (convolutionnal network vs multi-layers perceptron) and their exact learning criterion (pairwise comparison vs relative comparison).

Chapter 5

Bounds on the classification error

The representation learning algorithm introduced in Chapter 4 seeks at projecting the data in a representation space satisfying some labels and metric dependent constraints. The graph representation of the data built in the representation space learned by the introduced *MDRL* algorithm, following the method defined in Section 4.4, is expected to be a homophilic structure. If the graph is homophilic, graph-based classification algorithms are expected to perform optimally. In this chapter, we first show that a homophilic graph does not necessarily lead to an optimal classification through the label propagation algorithm introduced in Section 3.3. Non-triviality of linking a graph-based classification error to the data representation space or to a graph structure is consequently discussed.

Considering a mapping function learned via the introduced representation learning algorithm, to which extent the graph-based classification optimality can be ensured? In the following, some theoretical guarantees on the graph-based classification performance are to be defined, determined by the representation learned through our metric driven representation learning algorithm. Different lemmas and theorems will then be stated in order to perform a theoretical analysis of the classification framework introduced in the present thesis. Each of the theoretical elements will finally be proved in a last part.

5.1 Motivations

Based on the vectorial representation of the data learned by the *MDRL* algorithm, the objective is to bound the label propagation classification error that can be expected. Let d be a distance, $X = L \cup U$ be the dataset, $\mathbf{y_L}$ be the labels vector of instances from L and let us define T as in Equation 4.4. To apply the label propagation algorithm, a graph representation G of the data is built in the representation space mapped by ϕ. With our algorithm, the representation $\phi(x_i)$ of an instance $x_i \in X$ is learned to satisfy smooth-

ness related constraints, i.e. for each triplet $(x_i, x_j, x_k) \in T$, i.e. such that $y_i = y_j \neq y_k$, $d(\phi(x_i), \phi(x_j)) < d(\phi(x_i), \phi(x_k))$. Considering a perfectly learned representation space, each instance of X is supposed to be closer to each similarly labeled instance than to any dissimilarly labeled instances. In such a representation space, instances are grouped in as many dense clusters as there are different labels, allowing classification algorithms to optimally solve the task. Building a graph representation of the dataset allows us to relax the strong constraints on the representation space, as graph-based algorithms take advantage of the highlighted structure to solve the task. In graph-based algorithms, each instance does not need to be closer to each similarly labeled instances than dissimilarly labeled instances. Graph-based algorithms only require each instance to be locally closer to some similarly labeled instances than dissimilarly labeled instances. It intuitively seems that a graph-based classification algorithm applied on a graph built in a perfectly separated representation space will perform optimally and a zero error classification is expected from graphs built in such a representation space. It however appears that a a graph satisfying all the defined smoothness related constraints can lead to an inaccurate classification.

Despite our intuition, a graph representation of a dataset built in a representation space satisfying the smoothness related constraints can lead to an incorrect classification. In the following, we illustrate the fact that an incorrect classification can be obtained from a homophilic graph by introducing a simple example.

Let us consider a binary classification problem and the set of labeled examples $(x_0, 0), (x_1, 0), (x_2, 0), (x_3, 1), (x_4, 1)$. Let us hide the label of instance x_4 and let us define $X = L \cup U$ to be the dataset such that $L = \{x_0, x_1, x_2, x_3\}$ and $U = \{x_4\}$. The objective is thus to classify x_4, i.e. to predict the label associated with x_4. Based on Definition 4.4, let T be the set of triplets of instances expected to satisfy relative distances constraints based on L and $\mathbf{y_L}$. T is consequently composed of triplets (x_i, x_j, x_k) such that $y_i = y_j \neq y_k$. In the introduced context, the label set is only composed of two labels 0 and 1. The triplets (x_3, x_4, x_0) and (x_4, x_3, x_0) thus belongs to the triplet set T as well as all the possible compositions of the pair (x_3, x_4) with instances from class 0 and all the triplets (x_i, x_j, x_k) that can be computed where $y_i = y_j = 0$ and $x_k \in \{x_3, x_4\}$. The triplets (x_0, x_1, x_3) or (x_2, x_0, x_4) are other examples of the triplets composing T. Let us assume that from the representation of the instances, the following similarity matrix W was obtained:

$$\begin{pmatrix} 0 & 0.8 & 0.7 & 0.1 & 0.3 \\ 0.8 & 0 & 0.8 & 0.2 & 0.3 \\ 0.7 & 0.8 & 0 & 0.3 & 0.4 \\ 0.1 & 0.2 & 0.3 & 0 & 0.9 \\ 0.3 & 0.3 & 0.4 & 0.9 & 0 \end{pmatrix}$$

In the defined matrix, the red square wraps the links between the instances from the class 0 and the blue square surrounds the edges between the instances from the class 1. The row and columns highlighted in green correspond to the set of edges linking the testing instance x_4 to the rest of the graph. The weighted adjacency matrix W results in the graph represented in Figure 5.1. In Figure 5.1, width of the edges is representative of the instances closeness defined by the distance.

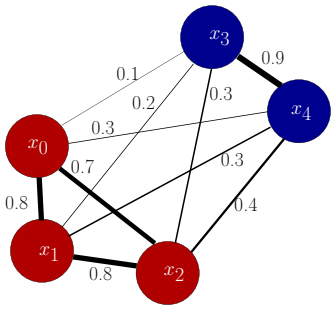


Figure 5.1 – Graph representation of the toy dataset

The matrix W can be seen as the adjacency matrix of a complete graph computed from X. To apply the label propagation algorithm introduced in Section 3.3, let us compute a transition matrix Q from G by row-normalizing W:

$$\begin{pmatrix} 0 & 0.42 & 0.37 & 0.05 & 0.16 \\ 0.38 & 0 & 0.38 & 0.1 & 0.14 \\ 0.32 & 0.36 & 0 & 0.14 & 0.18 \\ 0.07 & 0.13 & 0.2 & 0 & 0.6 \\ 0.16 & 0.16 & 0.21 & 0.47 & 0 \end{pmatrix}$$

It can be seen that in both the initial similarity matrix W or its row-normalized version, all the relative constraints associated with triplets from set T are satisfied. The complete graph underlying W consequently satisfies the smoothness assumption.

Based on the defined setting, let us unfold the label propagation algorithm computation introduced in Section 3.3 in order to classify x_4. Let us define the initial classes probabilities matrix

$$F^0 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0.5 & 0.5 \end{pmatrix}$$

At each step of the label propagation algorithm, the probabilities distributions are updated through the update rule $F^{t+1} = QF^t$. In our case, the probabilities distributions are updated as follows:

$$F^1 = \begin{pmatrix} 0.87 & 0.13 \\ 0.83 & 0.17 \\ 0.77 & 0.23 \\ 0.7 & 0.3 \\ 0.53 & 0.47 \end{pmatrix}$$

On the introduced simple problem with only one unlabeled instance, the label propagation algorithm converges in one iteration. The label c predicted by the label propagation algorithm is given by

$$c = \arg\max_{k \in \{0,1\}} F^1_{4k}$$

From F^1, the label predicted by the algorithm for x_4 is consequently 0, which is incorrect.

Let us try to understand why the label propagation gets the wrong answer even though similarly labeled instances are closer from each other than to dissimilarly labeled instances, i.e. the graph satisfies the smoothness assumption. A first observation that can be made is that the classes distribution of our labeled set is imbalance, and the class 0 is over-represented compared to the other class. A second observation that can be made concerns the adjacency matrix W. It can indeed be noticed that the global connectivity of x_4 to instances of the class 0 is higher than its connectivity to the class 1. The imbalance of the classes in the labeled set is thus not counterbalanced by the connectivity of x_4 to its true class.

It can be observed that the relevant information, i.e. connectivity of x_4 to similarly labeled instance x_3, was drowned by the numerous influences of dissimilarly labeled instances. From the introduced example, it can be seen that even if a graph satisfies the smoothness assumption, where similarly labeled instances are closer together than to dissimilarly labeled instances, the label propagation algorithm is not guaranteed to perform optimally. Satisfying the smoothness assumption does not ensure an optimal label propagation.

It should be noted that the latest observation is similar when the learned representation of the dataset not only satisfies the set of relative constraint but secures a fixed margin μ within the constraint, i.e. $\forall(x_i, x_j, x_k) \in T$, $d(\phi(x_i), \phi(x_j)) + \mu < d(\phi(x_i), \phi(x_k))$. The margin only postpones the issue contrasting the class imbalance and the connectivity influence problems.

Is pruning a solution?

Our previous example considered a complete graph representation of the dataset. It can be suggested that ignoring a few edges may solve the connectivity problem, by removing irrelevant edges between the two different classes. As seen in Section 3.2, neglecting some edges is the objective of pruning methods. Depending on the semantic of an edge, different simplification methods have been defined and can be applied. The different pruning methods can lead to various graph representations. Due to the various graph representations that can be obtained, the used pruning method influences the task computation (cf Section 3.4).

Let us recall the previously introduced adjacency matrix W of the complete graph from X and let us compare different graph representations that can be obtained by pruning it.

$$\begin{pmatrix} 0 & 0.8 & 0.7 & 0.1 & 0.3 \\ 0.8 & 0 & 0.8 & 0.2 & 0.3 \\ 0.7 & 0.8 & 0 & 0.3 & 0.4 \\ 0.1 & 0.2 & 0.3 & 0 & 0.9 \\ 0.3 & 0.3 & 0.4 & 0.9 & 0 \end{pmatrix}$$

Let us first compare different ϵ simplification, depending on the threshold ϵ (Table 5.1). With an ϵ simplification, the instances connected together through the remaining edges are ensured to have a minimum proximity depending on the threshold, as edges are kept depending on the set threshold. The ϵ simplification cannot guarantee anything about instances connectivity and can lead to disconnected instances. The main drawback of the ϵ simplification is the ϵ threshold tuning, which has a high influence on the obtained graph structure. When the chosen ϵ is too small ($\epsilon \in]0.2, 0.3[$ in our example), the risk is to keep most of the unwanted edges, which do not allows us to solve our class imbalance versus connectivity influence problem. On the opposite, relevant edges can be removed with a too restrictive ϵ ($\epsilon > 0.9$. in our example), which can lead to numerous disconnected instances. Disconnected instances can only be classified at random by the label propagation algorithm.

$\epsilon \in]0.2, 0.3[$	$\epsilon \in]0.4, 0.7[$	$\epsilon > 0.9$
$\begin{pmatrix} 0 & 0.8 & 0.7 & 0 & 0.3 \\ 0.8 & 0 & 0.8 & 0 & 0.3 \\ 0.7 & 0.8 & 0 & 0.3 & 0.4 \\ 0 & 0 & 0.3 & 0 & 0.9 \\ 0.3 & 0.3 & 0.4 & 0.9 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0.8 & 0.7 & 0 & 0 \\ 0.8 & 0 & 0.8 & 0 & 0 \\ 0.7 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9 \\ 0 & 0 & 0 & 0.9 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Table 5.1 – Possible obtained adjacency matrices W depending on the ϵ choice.

While the ϵ simplification removes edges depending on the absolute importance of each edge, k-nn simplification policy involves the relative importance of each edge of the neighbourhood of each instance. Mutual and symmetric simplifications lead to different guarantees on the graph. Due to its reciprocity constraint, the mutual k-nn simplification gives an upper bound on the neighbourhood size of each instance, while the symmetric k-nn simplification ensures a lower bound on the instances neighbourhood size. For our toy dataset, the graphs obtained through the two methods are a bit different but would lead to the same classification result based on the label propagation algorithm (Table 5.2).

	$k = 1$	$k = 2$
symmetric k-nn	$\begin{pmatrix} 0 & 0.8 & 0 & 0 & 0 \\ 0.8 & 0 & 0.8 & 0 & 0 \\ 0 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9 \\ 0 & 0 & 0 & 0.9 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0.8 & 0.7 & 0 & 0 \\ 0.8 & 0 & 0.8 & 0 & 0 \\ 0.7 & 0.8 & 0 & 0.3 & 0.4 \\ 0 & 0 & 0.3 & 0 & 0.9 \\ 0 & 0 & 0.4 & 0.9 & 0 \end{pmatrix}$
mutual k-nn	$\begin{pmatrix} 0 & 0.8 & 0 & 0 & 0 \\ 0.8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9 \\ 0 & 0 & 0 & 0.9 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0.8 & 0.7 & 0 & 0 \\ 0.8 & 0 & 0.8 & 0 & 0 \\ 0.7 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9 \\ 0 & 0 & 0 & 0.9 & 0 \end{pmatrix}$

Table 5.2 – Possible obtained adjacency matrices W depending on the k-nn simplification.

On the toy example, some of the different graphs obtained from various pruning methods allow the label propagation algorithm to perform optimally. While some of the pruned graphs do not perfectly separate the different classes, the other do. Different graph structures and properties can thus lead

to a similar classification. Pruning methods that are optimal for some datasets can be sub-optimal for others datasets. Depending on the initial graph, and a specified parameter, pruning can remove or keep too many edges. A common drawback shared by all the introduced pruning methods is that few properties of the graph can be ensured, in particular the graph connectivity. The label propagation final results being dependent on the graph structure and intern connectivity, defining some theoretical guarantees on the classification error obtained following the representation learning algorithm is difficult.

In this work, a classification task is aimed at being solved by applying the label propagation algorithm (cf Section 3.3) on a graph representation of the data. The graph representation used to solve the task is built in a representation space learned through our metric driven representation learning algorithm (cf Section 4.2). One of our goal is to bound the label propagation error that can be expected in the obtained graph representation.

Considering the influence of the pruning methods on the graph-based classification algorithms, being able to evaluate the classification errors calls for being able to characterized the pruned graph. The obtained graph is dependent on both the pruning method and the initial graph that has been simplified. The impact of the simplification step on the graph properties and structure can be difficult to characterize. The initial graph characteristics are subordinated to the representation space of the data and the function used to weight the edges. Due to all the defined uncertainties, a bound on the accuracy of the label propagation algorithm introduced in Section 3.3 depending on the representation of the data is not necessarly ensured.

Label propagation error can hardly be evaluated based on a specified graph. Label propagation is however guaranteed to be optimal if there are no edges between instances from different classes. All unlabeled instances must also be connected to labeled instances. Optimal label propagation can be obtained if the classes are well separated in the graph. A graph composed of as many connected components, homogeneous considering the instances labels, as classes will thus ensure an optimal label propagation.

Based on the latest observation, our theoretical analysis will not aim at bounding the classification error based on the representation learning algorithm error but at defining an initial context which provides an optimal label propagation. More precisely, we will seek at defining an initial context such that the graph build in the representation space learned by our representation learning algorithm separates the different classes. We claim that under some initial assumptions, the algorithm described in Section 4.2 provides a graph representation of the data that is optimal for classification through the label propagation algorithm.

5.2 Theoretical guarantees

In the following, we substantiate our claim by showing that a specific graph composed of c connected components homogeneous in class can be built. More precisely, a specific value of ϵ will be defined to apply an ϵ simplification on the complete graph representation of the dataset built in the representation space learned through our representation learning algorithm. We show that the existence of such an ϵ can be ensured by supposing that each unlabeled instance lies in a close neighourhood of at least one similarly labeled instance. The radius of the unlabeled instances neighbourhood will be defined depending on the representation space mapped by ϕ and the labeled instances.

To prove the existence of such an ϵ, the distance between two instances in the representation space mapped by ϕ is first shown to be bounded depending on their distance in the initial space and the learned mapping function (Lemma 1). It can simultaneously be shown that the relative constraint associated with a triplet of instances is satisfied if each of the triplets instances is sufficiently close, through the ϕ mapping, to a similarly labeled instance of any triplet satisfying its associated constraint (Lemma 2). The combination of the two previous lemmas allows us to define the necessary conditions such that non-labeled triplets respect their associated relative constraints in the representation space mapped by ϕ (Lemma 3). Based on Lemma 3, an ϵ allowing us to compute an optimal ϵ-graph for the label propagation algorithm can finally be exhibited (Proposition 1 and Theorem 1).

To introduce our propositions, let us first introduce some notations. Let us consider a multi-class classification context with the predefined set of labels $\mathcal{Y} = \{1, ..., c\}$. Let $D = \{(x_0, y_0), ..., (x_n, y_n)\}$ be a set of labeled examples, such that $\forall i \in \{0, ..., n\}$, $x_i \in \mathbb{R}^p$ and $y_i \in \mathcal{Y}$. From the initial set of labeled examples, let us define T as the set of triplets as defined in Equation 4.4. Let us define $L = \{x_0, ..., x_{l-1}\}$ the set of labeled instances and $U = \{x_l, ..., x_n\}$ the set of unlabeled instances. The dataset X is the set of instances $X = L \cup U$. From L, the subset $T_L \subset T$ of triplets can be defined as $T_L = \{(x_i, x_j, x_k) | (x_i, x_j, x_k) \in T \wedge x_i, x_j, x_k \in L\}$. Let T_U be defined as the other triplets, i.e. such that $T = T_L \cup T_U$.

Let Φ be the class of multi-layered perceptrons with a defined architecture such that $\forall \phi^m \in \Phi$, $\forall m \in \mathbb{N}$, it can be recursively defined succh that

$$\forall h \in \{0..m\}, \tilde{\phi}^h(x) = B^h \tanh(A^h \tilde{\phi}^{h-1}(x) + \alpha^h) + \beta^h$$

where for each layer h, A^h and B^h are the weights matrices parametrizing the neural network and α^h and β^h are the bias vectors associated with the layer.

Let ϕ^m be such that

$$\phi^m = \arg\min_{\tilde{\phi}^m} C(\tilde{\phi}^m | T_L)$$

as defined in Equation 4.5. ϕ^m is the mapping function from $\mathbb{R}^p$ to a new representation space $\mathbb{R}^q$, learned through our representation learning algorithm based on the set of triplets T_L (cf Section 4.2). Let us define the distance d to be either the Euclidean distance on $\mathbb{R}^p$ or on $\mathbb{R}^q$, depending on the context. Let us finally define W to be the adjacency matrix based on the pairwise Euclidean distance between the instances from X in the representation space mapped by ϕ, i.e. $\forall x_i, x_j \in X, W_{ij} = d(\phi^m(x_i), \phi^m(x_j))$. For the rest of the paper, for a matrix $A \in \mathbb{R}^{u \times v}$, let us define $A_{i:} = \sum_{j=1}^{v} A_{ij}$ and $\|A\|_1 = \sum_i A_{ji}$ is the $l1$-norm of the matrix A.

Let us now suppose that ϕ^m is learned such that $C(\phi^m | T_L) = 0$. The learned ϕ is a mapping function to a representation space in which the relative constraint associated with each triplet from T_L is satisfied. In the representation space mapped by ϕ, each labeled intance is closer to any similarly labeled instance than to any dissimilarly labeled instance. The different classes are pushed away from each other. Let us define

$$\Delta^+_{\phi^m} = \max_{(x_i,x_j,x_k) \in T_L} d(\phi^m(x_i), \phi^m(x_j))$$

$$(\text{respectively } \Delta^-_{\phi^m} = \min_{(x_i,x_j,x_k) \in T_L} d(\phi^m(x_i), \phi^m(x_k)))$$

the largest distance between two similarly labeled intances in the new representation space (respectively the smallest distance between dissimilarly labeled instances through the learned mapping function).

Now the notations and notions are defined, let us introduce our first lemma. The intent of the next lemma is to bound the distance between two instances in the representation space associated with ϕ depending on the distance between the two instances in the initial representation space.

Lemma 1. *Let us consider $\eta > 0$ and $x_i, x_j \in X$. If $d(x_i, x_j) < \eta$, then*

$$d(\phi^m(x_i), \phi^m(x_j)) < \eta \sqrt{\Pi_{h=0}^m \left[\sum_i \left(\sum_j |B_{ij}^h| \|A_{j:}^h\|_1 \right)^2 \right]}$$

The proof for the lemma (cf Section 5.3) simply bounds the effect of the various layers of the multi-layered perceptron.

For the next lemma, let us consider the instances in the new representation space. Lemma 2 shows that if the components of a triplet t from T lie in a sufficiently small radius around similarly labeled components of a triplet from T_L, the relative constraint associated with the triplet t is satisfied.

Lemma 2. *Let us consider* $\rho > 0$. *Let also consider* $(x_i, x_j, x_k) \in T_L$ *and* $(x_t, x_u, x_v) \in T$ *such that* $d(\phi^m(x_i), \phi^m(x_t)) < \rho$, $d(\phi^m(x_j), \phi^m(x_u)) < \rho$, $d(\phi^m(x_k), \phi^m(x_v)) < \rho$ *and* $y_i = y_t$, $y_j = y_u$ *and* $y_k = y_v$.

If $d(\phi^m(x_i), \phi^m(x_k)) - d(\phi^m(x_i), \phi^m(x_j)) > 4\rho$ *then*

$$d(\phi^m(x_t), \phi^m(x_v)) > d(\phi^m(x_t), \phi^m(x_u))$$

The introduced lemma can be proved (cf Section 5.3) by bounding the distances between the instances of triplet $(x_t, x_u, x_v) \in T$ — $d(\phi^m(x_t), \phi^m(x_u))$ and $d(\phi^m(x_t), \phi^m(x_v))$ — based on the distances between the components of the labeled triplet — $d(\phi^m(x_i), \phi^m(x_j))$ and $d(\phi^m(x_i), \phi^m(x_k))$ —, in the representation space mapped by ϕ. A condition on ρ, depending on the distances between the components of the labeled triplet in the representation space mapped by ϕ, can then be expressed.

By combining the two first lemmas, let us introduce a lemma on the generalization properties of our representation learning algorithm:

Lemma 3. *Let us consider* $(x_i, x_j, x_k) \in T_L$. $\exists \eta > 0$ *such that* $\forall (x_t, x_u, x_v) \in T_U$ *such that* $y_i = y_t$, $y_j = y_u$, $y_k = y_v$, $d(x_i, x_t) < \eta$, $d(x_j, x_u) < \eta$ *and* $d(x_k, x_v) < \eta$, *then*

$$d(\phi^m(x_t), \phi^m(x_v)) > d(\phi^m(x_t), \phi^m(x_u))$$

The lemma shows that the learning constraint is satisfied for all the triplets that are close enough from labeled instances. Its proof (cf Section 5.3) is based on the combination of Lemmas 1 and 2. It is shown that if the corresponding components of two triplets are close enough in the initial space and if components of one of the two considered triplets are well separated depending on their labels in the representation space mapped by ϕ, then components of the other triplets will also be well separated in the projection space considering their labels.

Let us recall the notation

$$\Delta^+_{\phi^m} = \max_{(x_i, x_j, x_k) \in T_L} d(\phi^m(x_i), \phi^m(x_j))$$

$$\text{(respectively } \Delta^-_{\phi^m} = \min_{(x_i, x_j, x_k) \in T_L} d(\phi^m(x_i), \phi^m(x_k))\text{)}$$

the largest distance between training instances of the same class in the projection space (respectively the smallest distance between training examples

of different classes in the projection space). From learning constraints, we know that $\Delta^{-}_{\phi^m} > \Delta^{+}_{\phi^m}$. Let us define $\hat{W}$, the ϵ-graph simplification of W

$$\hat{W}_{ij} = \begin{cases} W_{ij} & \text{if } W_{ij} < \epsilon \\ 0 & \text{otherwise} \end{cases}$$

with $\epsilon = \frac{\Delta^{-}_{\phi^m} + \Delta^{+}_{\phi^m}}{2}$.

Based on the previous lemmas, we state that the connected components of $\hat{W}$ correspond to the different classes under some conditions on the initial space:

Proposition 1. *We can define $\eta > 0$ such that if $\forall (x_t, x_u, x_v) \in T_U$, $\exists (x_i, x_j, x_k) \in T_L$ such that $d(x_i, x_t) < \eta$, $d(x_j, x_u) < \eta$, $d(x_k, x_v) < \eta$, $y_i = y_t$, $y_j = y_u$ and $y_k = y_v$, then :*

$\forall x_i, x_j \in X$, then

$$\hat{W}_{ij} = \begin{cases} W_{ij} & \text{if } y_i = y_j \\ 0 & \text{if } y_i \neq y_j \end{cases}$$

The proof of the proposition (cf Section 5.3) is two folds. Concerning pair of instances seen during the training step, the proof is based on the definition of ϵ and on the definition of ϕ. For other pairs of instances, the proposition is based on Lemma 1. By bounding W_{ij} depending on $d(\phi^m(x_i), \phi^m(x_j))$ (or $d(\phi^m(x_i), \phi^m(x_k))$ if $y_i \neq y_j$), η can indeed be defined depending on the non-linear transformation, $\Delta^{+}_{\phi^m}$ and on $\Delta^{-}_{\phi^m}$ such that $\hat{W}$ is composed of c connected components homogeneously labeled.

Let $LP(x_i)$ be the predicted label for instance x_i through the label propagation algorithm. Our main theorem can finally be introduced. The theorem claims the relevance of the representation space learned by the *MDRL* algorithm for the label propagation algorithm:

Theorem 1. *We can define $\eta > 0$ such that if $\forall (x_t, x_u, x_v) \in T_U$, $\exists (x_i, x_j, x_k) \in T_L$ such that $d(x_i, x_t) < \eta$, $d(x_j, x_u) < \eta$, $d(x_k, x_v) < \eta$, $y_i = y_t$, $y_j = y_u$ and $y_k = y_v$ then*

$$\forall (x_i, y_i) \in D | x_i \in U, LP(x_i) = y_i$$

Based on the strong assumption that each unlabeled instance is close from a distance η of at least one similarly labeled instance, the introduced theorem claims that in the representation space mapped by ϕ, a graph can be built such that the label propagation is optimal. The proof of the theorem is based on the Proposition 1 which allows us to build an ϵ-graph representation $\hat{W}$ of X where the connected components are homogeneous considering the instances labels (cf Section 5.3).

5.3 Proof

For the following, we first need to introduce a first lemma:

Lemma 4. *Let us define u,$v \in \mathbb{R}$ and $\nu > 0$. If $|u - v| < \nu$ then*

$$|\tanh(u) - \tanh(v)| < \nu$$

Proof. Let us define u,$v \in \mathbb{R}$ and $\nu > 0$ such that $|u - v| < \nu$. Let us suppose that $u > 0$ and $v < 0$, then $\tanh(u) < u$ and $\tanh(v) > v$. Thus $|\tanh(u) - \tanh(v)| < |u - v|$.

Now suppose that $u > 0$ and $v > 0$ such that $u \leq v$. As tanh is concave for $x > 0$, then $\tanh(v) \leq \tanh^{'}(u)(v - u) + \tanh(u)$. Then we have:

$$\begin{aligned} |\tanh(u) - \tanh(v)| &\leq |-\tanh^{'}(u)(v - u)| \\ &\leq |u - v| \\ &< \nu \end{aligned}$$

The case where $u > 0$, $v > 0$ and $u \geq v$ can be shown similarly, like the cases where $u < 0$ and $v < 0$.

Thus,

$$\forall u, v \text{ such that } |u - v| < \nu \text{ then } |\tanh(u) - \tanh(v)| < \nu$$

□

Lemma 1

Proof. Let us prove the lemma by recursion.

Let x_u and x_v be such that $d(x_u, x_v) < \eta$. Obviously $\forall i \in [1, d] |x_{v_i} - x_{v_i}| < \eta$. Let us evaluate $d(\phi^0(x_u), \phi^0(x_v))$.

$$\begin{aligned} d(\phi^0(x_u), \phi^0(x_v)) &= d(B^0 \tanh(A^0 x_u + \alpha^0) + \beta^0, B^0 \tanh(A^0 x_v + \alpha^0) + \beta^0) \\ &= \sqrt{\sum_i ((B^0 \tanh(A^0 x_u + \alpha^0)))_i - (B^0 \tanh(A^0 x_v + \alpha^0)))_i)^2} \quad \text{(definition of } d\text{)} \\ &= \sqrt{\sum_i (\sum_j B^0_{ij} (\tanh(A^0 x_u + \alpha^0)_j - \tanh(A^0 x_v + \alpha^0)_j))^2} \\ &\leq \sqrt{\sum_i (\sum_j |B^0_{ij} (\tanh(A^0 x_u + \alpha^0)_j - \tanh(A^0 x_v + \alpha^0)_j)|)^2} \end{aligned}$$

By definition of the absolute value, the following inequation holds:

$$d(\phi^0(x_u), \phi^0(x_v)) \leq \sqrt{\sum_i (\sum_j |B^0_{ij}| |(\tanh(A^0 x_u + \alpha^0)_j - \tanh(A^0 x_v + \alpha^0)_j)|)^2}$$

It can also be stated that

$$\begin{aligned}|(A^0x_u+\alpha)_j-(A^0x_v+\alpha)_j| &= |\sum_i A^0_{ji}(x_{u_i}-x_{v_i})|\\ &\leq \sum_i |A^0_{ji}||x_{u_i}-x_{v_i}|\\ &< \eta\sum_i |A^0_{ji}|\end{aligned}$$

Using Lemma 4,

$$\begin{aligned}|\tanh((A^0x_u+\alpha)_j)-\tanh((A^0x_v+\alpha)_j)| &< \eta\sum_i |A^0_{ji}|\\ &< \eta\|A^0_{j:}\|_1\end{aligned}$$

The distance $d(\phi^0(x_u),\phi^0(x_v))$ can consequently be bounded as follows:

$$\begin{aligned}d(\phi^0(x_u),\phi^0(x_v)) &< \sqrt{\sum_i(\sum_j |B^0_{ij}|\eta\|A^0_{j:}\|_1)^2}\\ &< \eta\sqrt{\sum_i(\sum_j |B^0_{ij}|\|A^0_{j:}\|_1)^2}\end{aligned}$$

Let us now suppose that $\exists\delta\in\mathbb{R}$ such that $d(\phi^m(x_u),\phi^m(x_v))<\delta$. Let us show that $d(\phi^{m+1}(x_u),\phi^{m+1}(x_v))<\delta\sqrt{\sum_i(\sum_j |B^{m+1}_{ij}|\|A^{m+1}_{j:}\|_1)^2}$.

$$\begin{aligned}d(\phi^{m+1}(x_u),&\phi^{m+1}(x_v)) = \sqrt{\sum_i(\phi^{m+1}(x_u)_i-\phi^{m+1}(x_v)_i)^2}\\ &= \sqrt{\sum_i(\sum_j B^{m+1}_{ij}(tanh(A^{m+1}\phi^m(x_u)+\alpha^{m+1})_j - tanh(A^{m+1}\phi^m(x_v)+\alpha^{m+1})_j)^2}\\ &\leq \sqrt{\sum_i(\sum_j |B^{m+1}_{ij}||tanh(A^{m+1}\phi^m(x_u)+\alpha^{m+1})_j - tanh(A^{m+1}\phi^m(x_v)+\alpha^{m+1})_j|)^2}\\ &\leq \sqrt{\sum_i(\sum_j |B^{m+1}_{ij}||(A^{m+1}\phi^m(x_u)+\alpha^{m+1})_j - (A^{m+1}\phi^m(x_v)+\alpha^{m+1})_j|)^2}\\ &\leq \sqrt{\sum_i(\sum_j |B^{m+1}_{ij}|\|A^{m+1}_{j:}\|_1|\phi^m(x_u)_j-\phi^m(x_v)_j|)^2}\end{aligned}$$

It has been seen that if $d(\phi^m(x_u),\phi^m(x_v))<\delta$ then $\forall j\in[1,q^m]$, $|\phi^m(x_u)_j-\phi^m(x_v)_j|<\delta$.

It follows that

$$d(\phi^{m+1}(x_u),\phi^{m+1}(x_v)) \leq \delta \sqrt{\sum_i (\sum_j |B_{ij}^{m+1}| \|A_{j:}^{m+1}\|_1)^2}$$

The recursion is consequently proved and so does the lemma. □

Lemma 2

Proof. Let us consider three instances x_i, x_j, $x_k \in X$. From the triangular inequality, we have that $d(x_i, x_j) \leq d(x_i, x_k) + d(x_k, x_j)$.

Applied on our triplets of instances:

$$\begin{aligned} d(\phi^m(x_i), \phi^m(x_k)) &\leq d(\phi^m(x_i), \phi^m(x_t)) + d(\phi^m(x_t), \phi^m(x_k)) \\ &\leq d(\phi^m(x_i), \phi^m(x_t)) + d(\phi^m(x_t), \phi^m(x_v)) + d(\phi^m(x_k), \phi^m(x_v)) \end{aligned}$$

and

$$\begin{aligned} d(\phi^m(x_t), \phi^m(x_v)) &\geq d(\phi^m(x_i), \phi^m(x_k)) - (d(\phi^m(x_t), \phi^m(x_i)) + d(\phi^m(x_k), \phi^m(x_v))) \\ &> d(\phi^m(x_i), \phi^m(x_k)) - 2\rho \end{aligned}$$

Similarly, we know that

$$\begin{aligned} d(\phi^m(x_t), \phi^m(x_u)) &\leq d(\phi^m(x_i), \phi^m(x_t)) + d(\phi^m(x_i), \phi^m(x_u)) \\ &\leq d(\phi^m(x_i), \phi^m(x_t)) + d(\phi^m(x_i), \phi^m(x_j)) + d(\phi^m(x_j), \phi^m(x_u)) \\ &< d(\phi^m(x_i), \phi^m(x_j)) + 2\rho \end{aligned}$$

Consequently

$$d(\phi^m(x_t), \phi^m(x_v)) - d(\phi^m(x_t), \phi^m(x_u)) > [d(\phi^m(x_i), \phi^m(x_k)) - d(\phi^m(x_i), \phi^m(x_j))] - 4\rho$$

From the previous analysis, for $d(\phi^m(x_i), \phi^m(x_k)) - d(\phi^m(x_i), \phi^m(x_j)) \geq 4\rho$, we have that $d(\phi^m(x_t), \phi^m(x_v)) > d(\phi^m(x_t), \phi^m(x_u))$. □

Lemma 3

Proof. Let us consider $\eta > 0$ and a triplet $(x_t, x_u, x_v) \in T_U$ such that $y_i = y_t = y_j = y_u \neq y_k = y_v$ and $d(x_i, x_t), d(x_j, x_u), d(x_k, x_v) < \eta$. By Lemma 1, we have that

$$\begin{cases} d(\phi^m(x_i), \phi^m(x_t)) < \eta \sqrt{\Pi_{h=1}^m [\sum_i (\sum_j |B_{ij}^h| \|A_{j:}^h\|_1)^2]} \\ d(\phi^m(x_j), \phi^m(x_u)) < \eta \sqrt{\Pi_{h=1}^m [\sum_i (\sum_j |B_{ij}^h| \|A_{j:}^h\|_1)^2]} \\ d(\phi^m(x_k), \phi^m(x_v)) < \eta \sqrt{\Pi_{h=1}^m [\sum_i (\sum_j |B_{ij}^h| \|A_{j:}^h\|_1)^2]} \end{cases}$$

By Lemma 2, if η is such that

$$d(\phi^m(x_i), \phi^m(x_k)) - d(\phi^m(x_i), \phi^m(x_j)) \geq 4\eta \sqrt{\Pi_{h=1}^m [\sum_i (\sum_j |B_{ij}^h| \|A_{j:}^h\|_1)^2]}$$

we have that

$$d(\phi^m(x_t), \phi^m(x_v)) > d(\phi^m(x_t), \phi^m(x_u))$$

Consequently for

$$\eta \leq \frac{d(\phi^m(x_i), \phi^m(x_k)) - d(\phi^m(x_i), \phi^m(x_j))}{4\sqrt{\Pi_{h=1}^m [\sum_i (\sum_j |B_{ij}^h| \|A_{j:}^h\|_1)^2]}}$$

if $d(x_i, x_t), d(x_j, x_u), d(x_k, x_v) < \eta$, then

$$d(\phi^m(x_t), \phi^m(x_v)) > d(\phi^m(x_t), \phi^m(x_u))$$

□

Proposition 1

Let us define $\hat{W}$, the ϵ-graph simplification of W

$$\hat{W}_{ij} = \begin{cases} W_{ij} & \text{if } W_{ij} < \epsilon \\ 0 & \text{otherwise} \end{cases}$$

where $\epsilon = \frac{\Delta_{\phi^m}^- + \Delta_{\phi^m}^+}{2}$.

Proof. Let us first consider the case where x_i and $x_j \in L$ and let us suppose $y_i = y_j$. By definition of $\Delta_{\phi^m}^+$, $d(\phi^m(x_i), \phi^m(x_j)) \leq \Delta_{\phi^m}^+$.

By definition of ϵ, $\epsilon = \frac{\Delta_{\phi^m}^+ + \Delta_{\phi^m}^-}{2}$. Consequently $\epsilon > \frac{\Delta_{\phi^m}^+ + \Delta_{\phi^m}^+}{2}$ and $\epsilon > d(\phi^m(x_i), \phi^m(x_j))$. We obtain that $\hat{W}_{ij} = W_{ij}$.

Let us now suppose that $y_i \neq y_j$. By definition of $\Delta_{\phi^m}^-$, $\epsilon < \Delta_{\phi^m}^-$ and $d(\phi^m(x_i), \phi^m(x_j)) \geq \Delta_{\phi^m}^-$. It follows that $d(\phi^m(x_i), \phi^m(x_j)) > \epsilon$ and $\hat{W}_{ij} = 0$.

Let us now consider the case where either x_i or $x_j \in U$. Let us first suppose that $y_i = y_j$. Let $x_l \in X$ be such that $y_l \neq y_i$. Let us consider $(x, x_+, x_-) \in T_L$ such that $d(x_i, x) < \eta$, $d(x_j, x_+)\eta$, $d(x_l, x_-) < \eta$, $y = y_i$ and $y_- = y_l$.

Thus by the triangular inequality:

$$\begin{aligned} W_{ij} &= d(\phi^m(x_i), \phi^m(x_j)) \\ &\leq d(\phi^m(x_i), \phi^m(x)) + d(\phi^m(x), \phi^m(x_j) \\ &\leq d(\phi^m(x_i), \phi^m(x)) + d(\phi^m(x), \phi^m(x_+) + d(\phi^m(x_j), \phi^m(x_+)) \end{aligned} \quad (5.1)$$

By Lemma 1, $d(\phi^m(x_i), \phi^m(x)) < \eta\sqrt{\Pi_{h=0}^{m}[\sum_i(\sum_j|B_{ij}^h|\|A_{j:}^h\|_1)^2]}$ and $d(\phi^m(x_j), \phi^m(x_+)) < \eta\sqrt{\Pi_{h=0}^{m}[\sum_i(\sum_j|B_{ij}^h|\|A_{j:}^h\|_1)^2]}$.

From Inequality 5.1, W_{ij} can thus be bounded as follows:

$$\begin{aligned} W_{ij} &< d(\phi^m(x), \phi^m(x_+)) + 2\eta\sqrt{\Pi_{h=0}^{m}[\sum_i(\sum_j|B_{ij}^h|\|A_{j:}^h\|_1)^2]} \\ &< \Delta_{\phi^m}^{+} + 2\eta\sqrt{\Pi_{h=0}^{m}[\sum_i(\sum_j|B_{ij}^h|\|A_{j:}^h\|_1)^2]} \end{aligned}$$

Similarly, let us suppose that $y_i \neq y_j$. Let $x_l \in X$ be such that $y_l = y_i$ and let us consider $(x, x_+, x_-) \in T_L$ such that $d(x_i, x) < \eta$, $d(x_j, x_-) < \eta$, $d(x_l, x_+) < \eta$, $y = y_i$ and $y_- = y_l$. Thus by the triangular inequality:

$$\begin{aligned} d(\phi^m(x), \phi^m(x_-)) &\leq d(\phi^m(x), \phi^m(x_i)) + d(\phi^m(x_i), \phi^m(x_-)) \\ &\leq d(\phi^m(x), \phi^m(x_i)) + d(\phi^m(x_i), \phi^m(x_j)) + d(\phi^m(x_j), \phi^m(x_-)) \end{aligned}$$

and

$$\begin{aligned} W_{ij} &= d(\phi^m(x_i), \phi^m(x_j)) \\ &\geq d(\phi^m(x), \phi^m(x_-)) - (d(\phi^m(x_j), \phi^m(x_-)) + d(\phi^m(x), \phi^m(x_i))) \end{aligned}$$

Based on Lemma 1, a lower bound for W_{ij} can be defined

$$\begin{aligned} W_{ij} &> d(\phi(x), \phi(x_-)) - 2\eta\sqrt{\Pi_{h=0}^{m}[\sum_i(\sum_j|B_{ij}^h|\|A_{j:}^h\|_1)^2]} \\ &> \Delta_{\phi}^{-} - 2\eta\sqrt{\Pi_{h=0}^{m}[\sum_i(\sum_j|B_{ij}^h|\|A_{j:}^h\|_1)^2]} \end{aligned}$$

Consequently, by defining $\eta \leq \frac{\Delta_{\phi^m}^{-} - \Delta_{\phi^m}^{+}}{4\sqrt{\Pi_{h=1}^{m}[\sum_i(\sum_j|B_{ij}^h|\|A_{j:}^h\|_1)^2]}}$ W is characterized as follows

$$W_{ij}\begin{cases} < \Delta_{\phi^m}^{+} + \frac{\Delta_{\phi^m}^{-} - \Delta_{\phi^m}^{+}}{2} = \epsilon & \text{if } y_i = y_j \\ > \Delta_{\phi^m}^{-} - \frac{\Delta_{\phi^m}^{-} - \Delta_{\phi^m}^{+}}{2} = \epsilon & \text{if } y_i \neq y_j \end{cases}$$

which concludes our proof. □

Theorem 1

Based on the Proposition 1, Theorem 1 claims that a threshold η can be defined such that the ϵ-graph built in the representation space mapped by ϕ with the threshold $\epsilon = \eta$ allows the label propagation algorithm to perform optimally.

Proof. Let us consider the ϵ-graph $\hat{W}$ with $\epsilon = \frac{\Delta^-_{\phi^m} + \Delta^+_{\phi^m}}{2}$. According to Proposition 1, the weight W_{ij} of every edge e_{ij} such that $y_i \neq y_j$ will be reduced to 0 and the weight of every intra-class edge remains non-negative. Thus the connected components remaining in the graph are only composed of similarly labeled instances and the label propagation algorithm will be optimal. □

Chapter 6

Empirical analysis

The objective of the introduced representation learning algorithm is to project the data in a representation space in which the classification task can be more optimally solved, compared to the initial representation space. More precisely, the data is projected in a representation space such that two similarly labeled instances are closer from each other than from a dissimilarly labeled instance. For the remainder of the present chapter, let us consider closeness to be expressed through the Euclidean distance. In this chapter, we aim at empirically evaluate the gain of the introduced representation algorithm. We first describe more precisely how the mapping function ϕ is learned by optimizing the cost function introduced in Section 4.2. The several datasets on which the experiments will be performed and the several (representation, distance) settings that will be compared in the experiments are then successively introduced.

As a first step of the empirical evaluation, capability of the proposed representation learning algorithm to meet the metric related constraints will first be empirically evaluated. More precisely, the satisfaction of the guiding constraints is being empirically evaluated for the different datasets in the different settings. A theoretical analysis has been performed in Chapter 5 and some theoretical guarantees on the classification task have been defined, based on strong assumptions on the initial representation space. In a second phase, satisfaction of the introduced assumptions is evaluated. Let us then focus on the evaluation of the empirical gain of the *MDRL* algorithm for classification task. A third step is consequently to analyse the influence of the representation space in which the data is projected by evaluating the improvement of the classification task compared to other approaches. Some open questions emerged from the theoretical and the empirical analysis. We will discuss them by empirically exploring some possible extensions of the work introduced in the present thesis.

6.1 MDRL - Triplet based neural network learning

Let us assume the same notation as previously introduced and let T be the set of triplets as introduced in Equation 4.4 based on L. To map the dataset X in a representation space satisfying a set of relative constraints based on the triplets set T and related to the smoothness assumption, the mapping function ϕ is learned to minimize the cost function $C(\phi|T)$ as defined in Equation 4.5:

$$C(\phi|T) = \sum_{(x_i,x_j,x_k)\in T} max(0, \mu - [d(\phi(x_i), \phi(x_j) - d(\phi(x_i), \phi(x_k))])$$

where μ is a set margin. In order to learn the multi-layered perceptron ϕ, a siamese neural network is created based on ϕ, computing the loss introduced in Equation 4.3 for any triplet $(x_i, x_j, x_k) \in T$:

$$max(0, \mu - [d(\phi(x_i), \phi(x_j) - d(\phi(x_i), \phi(x_k))])$$

The loss function is minimized by backpropagating the cost obtained for each triplet that is given in input to the siamese network. The weights updates due to backpropagation can be seen as part of an iterative gradient descent. A criterion musts be defined to stop the gradient descent. The gradient descent algorithm is usually stopped when the model converges. A model is said to be convergent when the cost function used to evaluate the model is stable, i.e. has a low variation from an epoch to another.

Gradient descent aims at minimizing the expected risk of the model through the minimization of the empirical risk. The empirical risk is minimized by updating the model depending on a selected example. Such a learning algorithm is however prone to overfitting. A model is said to overfit when it performs great on instances used for learning but has a bad generalization performance on unseen instances. The model can notably memorize some of the examples. Several solutions can be applied to avoid overfitting ([Bottou, 2012]).

The model can be given in input the same sequence of instances. To avoid overfitting, having predictible inputs musts be averted. To prevent cycles that can occur during the learning phase by selecting examples depending on a defined ranking, the set of triplets T is randomly browsed. Our cost function $C(\phi|T)$ is consequently minimized by stochastic gradient descent.

Model overfitting can be detected when the cost function evaluated on training instances is minimized but its value on unseen instances increases. To detect the simultaneous decrease of the objective function value on training examples and the increase of its value on unseen instances, let us split the set of instances used to train the model into two sets. One of the two created sets, called the validation set, is used as a sanity check, the other

set being named the training set. The increase of the cost function value on the validation set on the opposite to a decrease of the same value for the training set will consequently be the criterion used to stop the stochastic gradient descent used to learn our model.

Algorithm 4 summarises our approach, which we refer to as the **MDRL** algorithm, to learn an optimal representation for graph based classification.

Algorithm 4: MDRL: Representation learning algorithm for graph-based classification

Data: $X = L \cup U$ the dataset
$\mathbf{y_L}$ the labels vector for instances from L
ϕ a multi-layered perceptron
μ the margin > 0
d a distance
α the learning rate > 0
Result: ϕ the learned transformation function
`// Initialization`
$L_t, L_v \leftarrow$ the split of instances from L ;
$T \leftarrow$ the set of triplets based on L as defined in Equation 4.4;
$T_t \leftarrow$ the subset of triplets from T where instances are in L_t;
$T_v \leftarrow$ the subset of triplets from T where instances are in L_v;
`// Stochastic gradient descent`
while *not convergence of* $C(\phi|T_t)$ *&* *not increase of* $C(\phi|T_v)$ **do**
  $(x_i, x_j, x_k) \leftarrow$ random triplet sampled from T_T;
  $error \leftarrow max(0, \mu - [d(\phi(x_i), \phi(x_j) - d(\phi(x_i), \phi(x_k))])$;
  for *layer* $h \in \{0..m\}$ **do**
    for *weight* A_{ij}^h *in the parameters set of layer* h **do**
      $A_{ij}^h \leftarrow A_{ij}^h + \lambda \frac{\delta error}{\delta A_{ij}^h}$;
    end
  end
end

6.2 Datasets

Let us first introduce the different datasets on which the several experiments will be performed. All datasets characteristics are summarized in Table 6.1.

A first set of experiments are performed on variations of the artificial dataset *circle*.

The *circle* dataset is used to compare the targeted algorithms on a dataset where a classical distance like the Euclidean distance is at least locally representative of the task. The *circle* dataset is composed of two dimensional

Name	type	size	features	classes
circle	artificial	500	2	2
perturbedCircle1	artificial	500	4	2
perturbedCircle2	artificial	500	4	2
perturbedCircle3	artificial	500	6	2
cancer	real	683	9	2
ionosphere	real	351	33	2
vehicle	real	596	18	4
digit	real	1797	64	10

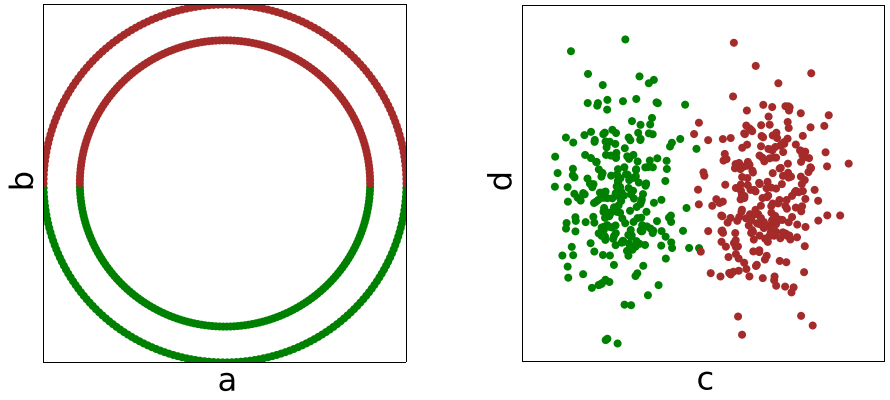
Table 6.1 – Summary of the different datasets characteristics

points $x = (a, b)$ that are sampled from two concentric circles (Figure 6.5a). For simplicity, let us assume that the circles are centered on the $(0, 0)$ point. The two features a and b of each point x of the dataset correspond to its euclidean coordinates. Each circle is associated with a class. The class of each point is consequently defined depending on the circle the point belongs to, i.e. on the value of $a^2 + b^2$. The two features characterizing an instance have an equal influence on the label, as the label of each instance x is defined depending on $a^2 + b^2$. Among all the possible combinations, the radius of the two circles were chosen such that for each instance, there exists instances lying on the same circle that are closer, considering a classical distance like the Euclidean distance, than any instance from the other circle. Considering both an ϵ-neighborhood or a k-nn neighbourhood, a close neighbourhood can be defined for each instance from the *circle* dataset such that it contains a majority of similarly labeled instances. The classical Euclidean distance, the distance considered in our setting, locally reflects the task similarity in the initial vectorial representation.

As previously seen, due to the dataset construction process, the Euclidean distance is locally representative of the task similarity for the *circle* dataset. The objective is to evaluate our algorithm for a dataset for which its initial vectorial representation is not representative of the task, considering the Euclidean distance. The following dataset aims at evaluating the ability of the several compared algorithms that will be introduced in Section 6.3 to extract the features related to the task while ignoring the irrelevant features.

In order to have a dataset for which the Euclidean distance is not representative of the task in its initial representation space, let us introduce some perturbations in the *circle* dataset. Let the dataset *perturbedCircle1* (Figure 6.5b) be a first perturbed version of the *circle* dataset. The vectorial representation of the dataset instances is composed of four numerical features, i.e. $x = (a, b, c, d)$. The two first features a and b of an instance x are sampled likewise to the *circle* dataset and the two added features c and d are

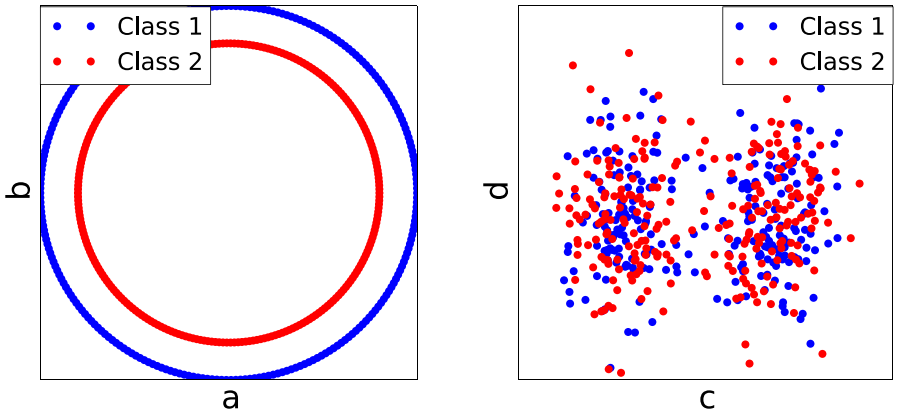
sampled from a two-dimensional gaussian. Similarly to *circle* dataset, the label of an instance from *perturbedCircle1* is defined depending on a^2+b^2, i.e. on the circle it lies on considering the two first features. Depending on the sign of b, the added features c and d are sampled from one of two well separated two-dimensional gaussian distributions. The relationships between the two groups of features are illustrated in Figure 6.1.



(a) The two first features (a, b) (b) The two last features (c, d)

Figure 6.1 – Visualisation of the relation between the two groups of features (a, b) and (c, d) of the *perturbedCircle1* dataset.

As the two first features and the label of any instance from *perturbedCircle1* dataset are based on the *circle* dataset, the Euclidean distance is still locally representative of the task similarity considering only the first two features a and b. Two close instances on a same circle but with opposite sign on the b-axis will be pretty different considering the c and d features. On the opposite, two instances lying on the two different circles but sharing the same sign on the b-axis will be considered as close depending on their c and d features. It can be observed in Figure 6.2 that instances of both classes are mixed considering the second group of features. The added two features c and d act as a kind of structured noise. As the Euclidean distance gives equal influence to all the features, closeness of two instances will equally depend on the circle they lie on and their memberships to the same gaussian, i.e. their signs on the b-axis. The addition of a structured noise can weaken the relevance of a classical distance, which gives equal importance to all the features. It can also be observed in the two-dimensional PCA projection of Figure 6.5b that the two classes are not well separated and the two initial circles are blured. The objective of the introduced dataset is to evaluate the ability of our algorithm to extract the relevant features for the task and to ignore irrelevant features.

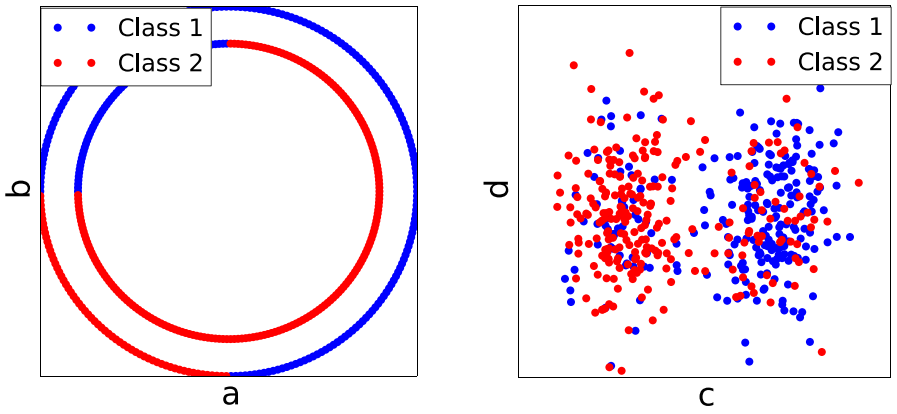


(a) The two first features (a, b) (b) The two last features (c, d)

Figure 6.2 – Visualisation of the classes distribution of *perturbedCircle1* dataset depending on the two groups of features (a, b) and (c, d).

In the previous dataset, the added features are not related to the task. The *perturbedCircle2* dataset that is now introduced aims at evaluating the ability of the several compared algorithms to take advantage of the labeling information contained in the different features. The *perturbedCircle2* dataset (Figure 6.5c) is another perturbed version of the *circle* dataset. Each instance of the *perturbedCircle2* dataset is composed of four numerical features, $x = (a, b, c, d)$, which are sampled following the *perturbedCircle1* dataset creation process. The described dataset is similar to *perturbedCircle1* dataset and only varies on the labeling process. Each instance label depends either on the circle it lies on if $a \geq 0$, or on the sign of b if $a < 0$ (Figure 6.3).

The *perturbedCircle2* dataset features are obtained from the same construction process than the *perturbedCircle1* dataset. Each feature thus has the same influence on the computed Euclidean distance. Both pairs of features are however not similarly related to the labeling of the *perturbedCircle2* dataset. Due to the labeling process change, the Euclidean distance may not be as locally representative of the task similarity than for the previous datasets. The added two features act as a competing explanation of the labels. The Euclidean distance relevance for the task has weakened with respect to the *circle* dataset, but not necessarily with respect to the *perturbedCircle1* dataset as the added features are somehow related to the task (Figure 6.3b), on the opposite to the structured noise added in the *perturbedCircle1* dataset (Figure 6.2b). The two gaussian distributions of the *perturbedCircle2* dataset are indeed more homogeneous, considering the classes of the instances, (Figure 6.3b) than the gaussian distributions from the *perturbedCircle1* dataset (Figure 6.2b). The objective of *perturbedCircle2* dataset is consequently



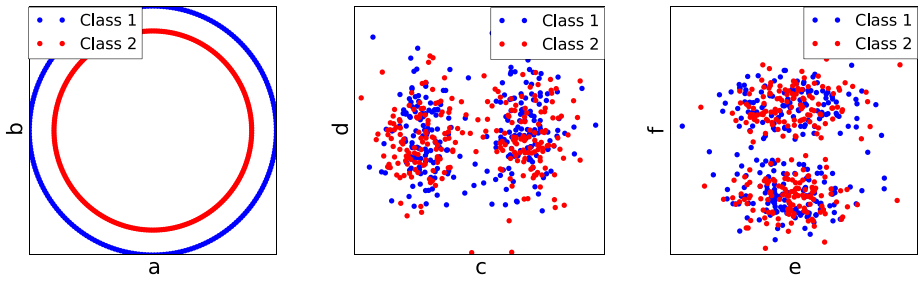
(a) The two first features (a, b) (b) The two last features (c, d)

Figure 6.3 – Visualisation of the classes distribution of the *perturbedCircle2* dataset depending on the two groups of features (a, b) and (c, d).

to evaluate the ability of our algorithm to extract the labeling information contained in the different features.

The last perturbed variant of the *circle* dataset that will be introduced is the *perturbedCircle3* dataset. Empirical experiments on the *perturbedCircle3* dataset aim at evaluating how the different algorithms take advantage of the informative features that are drown in a higher dimensional vectorial representation. More precisely, the *perturbedCircle3* dataset (Figure 6.5d) is a six-dimensional dataset, $x = (a, b, c, d, e, f)$, based on the *perturbedCircle1* dataset. The four first features sampling and the labeling processes are similar to the processes described for the *perturbedCircle1* dataset. The two last features, e and f, are sampled from two other well separated two-dimensional gaussian distributions depending on the sign of the feature a.

As it is based on the *perturbedCircle1* dataset, the Euclidean distance is relevant for the task considering only the two first features of any instance (Figure 6.4a). The four other features are pairs of coordinates sampled from different gaussian distributions, depending on the sign of either the first or the second feature. With the added two dimensional gaussian coordinates compared to the *perturbedCircle1* dataset, another structured perturbation which does not bring any information about the labeling is introduced (Figures 6.4b and 6.4c). The described dataset is a more perturbed version of the *perturbedCircle1* dataset. The introduced perturbation makes the Euclidean distance unable to represent the task similarity considering the four last features (Figure 6.5d). The relevant features carrying the labeling information are drowned in the pool of unrelated features, as the Euclidean distance gives



(a) The first pair of features (a, b)

(b) The second pair of features (c, d)

(c) The last pair of features (e, f)

Figure 6.4 – Visualisation of the classes distribution of *perturbedCircle3* dataset depending on the three groups of features (a, b), (c, d) and (e, f).

equal importance to all the features. The objective of the *perturbedCircle3* dataset is to evaluate the capability of our algorithm to extract the informative features in the bigger set of irrelevant features.

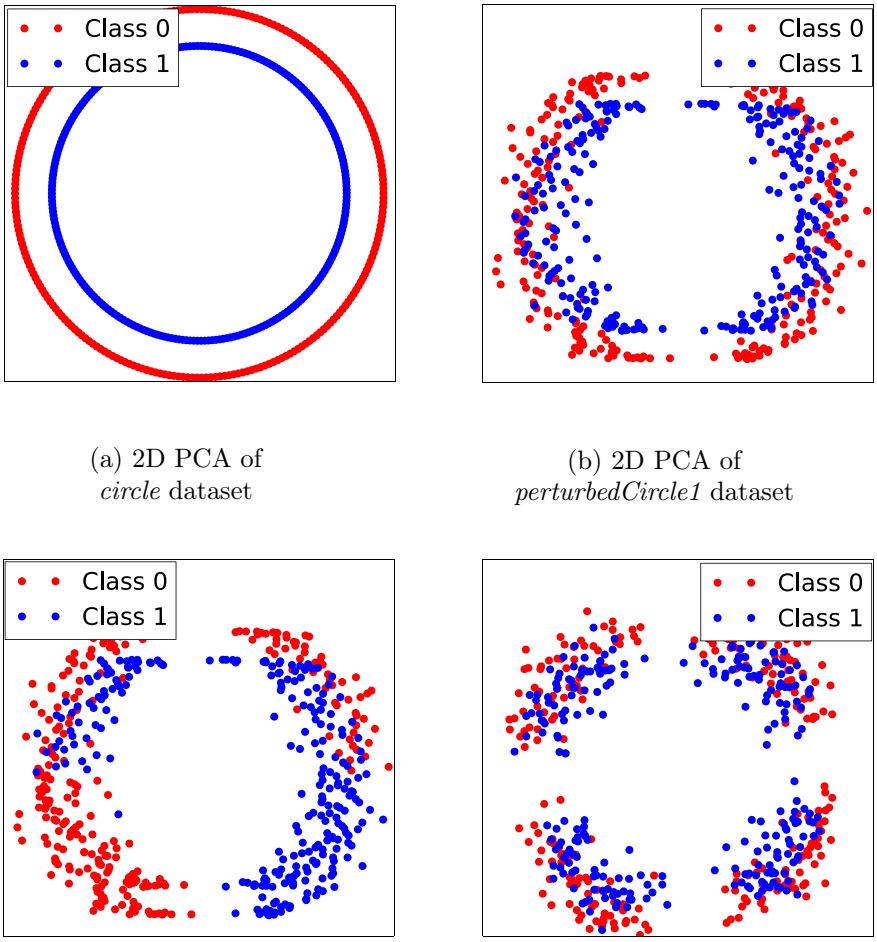
A main objective of the experimental section is to evaluate the applicability of the introduced approach for real world problems. Let us empirically evaluate our algorithm on real world datasets. As for artificial datasets, the objective is to evaluate the several algorithms on datasets with increasing complexity, considering the representation space complexity or the number of classes. Let us introduce the real world datasets on which experiments will be performed.

A first real world based dataset that will be used is the *cancer* dataset. The *cancer* dataset[1] is a dataset that associates with each instance a label indicating if a tumour is malignant or benign. *Cancer* dataset features are numerical attributes based on biological cell related measures. Instances with missing values have been removed. Based on the two-dimensional PCA projection of the dataset (Figure 6.7a), we can observe that the two classes of the *cancer* dataset seems almost separated. The Euclidean distance is probably representative of the task similarity in the initial representation space.

Ionosphere dataset[2] is another real world dataset that will be targeted, characterized by more features. The *ionosphere* dataset is composed of continuous features based on some physical measurements, with no missing val-

[1] http://archive.ics.uci.edu/ml/index.html, Breast Cancer Wisconsin (Original) Data Set

[2] http://archive.ics.uci.edu/ml/index.html, Ionosphere Data Set



(a) 2D PCA of *circle* dataset

(b) 2D PCA of *perturbedCircle1* dataset

(c) 2D PCA of *perturbedCircle2* dataset

(d) 2D PCA of *perturbedCircle3* dataset

Figure 6.5 – Two-dimensional PCA projections of the artifical datasets

ues. The binary labeling refers to the presence, or not, of some structures in the ionosphere. The task is consequently to predict the presence, or not, of those structures depending on new physical measurements. The two-dimensional PCA projection of the two classes of the dataset (Figure 6.7b) is less structured than the *cancer* dataset. The Euclidean distance in the initial representation space seems to be less relevant to solve the classification task than for the *cancer* dataset.

Either artificial or real, the datasets that were introduced until now were divided in two classes. Real world problems can however be more complex,

needing more than two classes. Let us introduce datasets with more classes.

The *vehicle* dataset[3] is a real world dataset composed of instances grouped into four different classes, corresponding to a type of vehicle the instance can be related to. Each instance is characterized by a set of numerical features based on the processing of a picture of a vehicle. Entries with missing values were removed. From the analysis of the two-dimensional PCA projection of the dataset (Figure 6.7c), the different classes do not appear to be well separated. Consequently, there is a concern about the Euclidean distance not being representative of the task similarity in the initial representation space of the *vehicle* dataset.

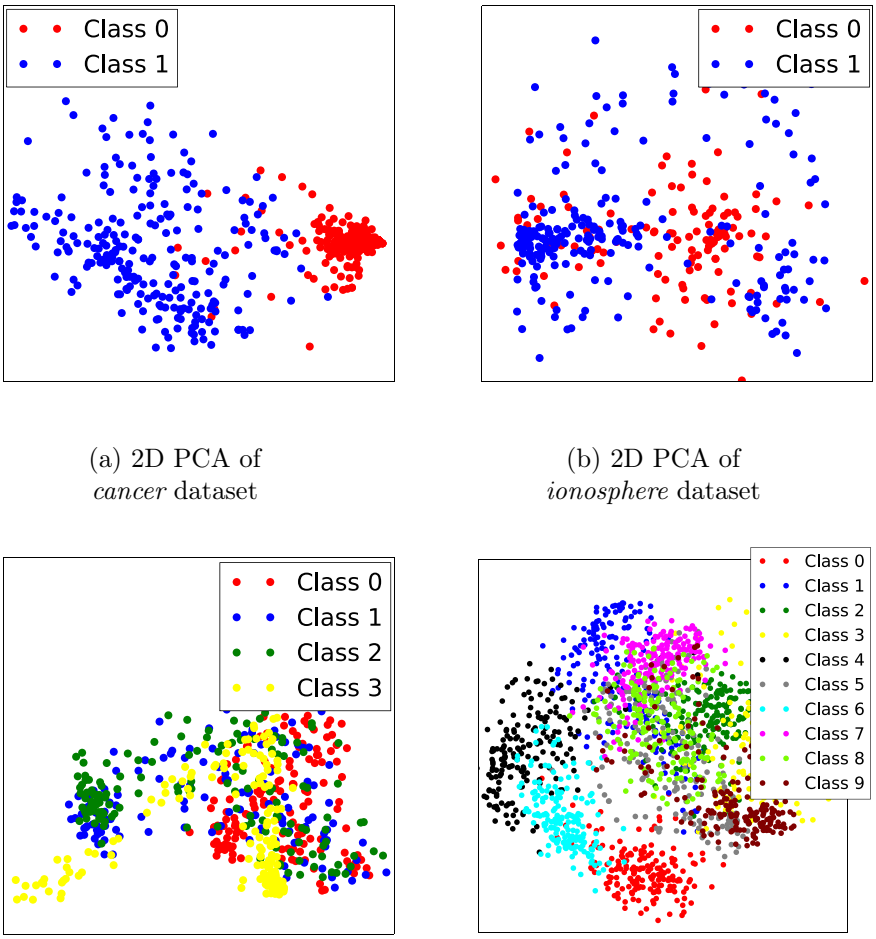
The last real world dataset that will be introduced is the *digit* dataset[4]. The *digit* dataset is composed of the flat vectorial representation of 8×8 images of hand-written digits. The label of each instance corresponds to the digit represented by the associated image, i.e. an integer from 0 to 9. In the two-dimensional PCA projection of the *digit* dataset (Figure 6.7d), most of the different classes seem to be grouped in dense area, although some are entirely overlapped by another class. There are also some overlaps between the borders of different classes. Even if the Euclidean distance seems to be mostly locally representative of the labels similarity, it do not seems to be globally adapted to the task.

6.3 Comparative settings

The objective is to evaluate the representation learning algorithm and the gain that can be expected for the classification task resolution. The setting that will be considered when evaluating our algorithm is the application of the Euclidean distance in the representation space mapped by the learned function ϕ. Let us refer to the described setting under the name *MDRL*. In order to empirically evaluate the adequacy of the Euclidean distance to the representation space in which the data is projected, let us compare the *MDRL* setting to other settings. Let us define the different pairs (representation space, distance) that will be applied on the different datasets in the present chapter. The several settings that will be compared are summarized in Table 6.2.

The first trivial baseline the setting *MDRL* will be compared to is the simple application of the Euclidean distance in the initial representation space of the considered dataset. The baseline setting will be refered as the

[3] http://archive.ics.uci.edu/ml/index.html, Statlog (Vehicle Silhouettes) Data Set
[4] The digit dataset from Scikit-learn, [Pedregosa et al., 2011]



(a) 2D PCA of *cancer* dataset

(b) 2D PCA of *ionosphere* dataset

(c) 2D PCA of *vehicle* dataset

(d) 2D PCA of *digits* dataset

Figure 6.6 – Two-dimensional PCA projections of the real world datasets

Setting	Representation space	distance
Initial space	$\mathcal{X}$	Euclidean
MDRL	$\phi(\mathcal{X})$	Euclidean
Feature selection	FS($\mathcal{X}$)	Euclidean
LMNN	$\mathcal{X}$	LMNN

Table 6.2 – Summary of the different settings characteristics

initial space setting.

The Euclidean distance can be irrelevant for solving the task for a dataset

because of some features composing the vectorial representation of the data that are unrelated to the task. Extracting the most relevant features of the dataset can consequently be helpful to solve the task. The second setting to which *MDRL* will be compared is a setting where the representation space is learned to be the more representative of the labeling. More specifically, the Euclidean distance will be applied on the vectorial representation computed from the more representative subset of the initial features. In the new vectorial representation of the data, only the most representative features of the initial representation space will be kept. By applying the tree construction method of the *Extra tree classifier* algorithm[5] ([Geurts et al., 2006], cf Section 2.4), a score is associated with each feature. A threshold is fixed and the subset of features with a score higher or equal than the threshold are kept to constitute the new vectorial representation of the instances. In our framework context, the threshold applied on the computed score to extract the most relevant features is the mean of all scores. The introduced setting will be refered as the *feature selection* setting.

On one hand, as previously discussed, the representation can be learned to be more adapted to the used distance. On the other hand, the representation can be fixed and an adapted distance for the representation space can be learned. The *MDRL* setting will be compared to a Mahalanobis distance learned through the well known LMNN metric learning algorithm[6] ([Weinberger and Saul, 2009], cf Section 2.4) applied in the initial representation space. Let us refer to the described setting as the *LMNN* setting.

6.4 Representation learning

The objective of the introduced representation learning algorithm is to learn a function ϕ mapping instances from the initial representation space to a representation space in which similarly labeled instances are close, depending on the selected metric, and dissimilarly labeled instances are pushed away. The mapping ϕ is learned in order to satisfy some triplet relative constraints.

More precisely, from one of the previously introduced datasets, let us consider to have a set of labeled examples $\{(x_0, y_0), ..., (x_{n-1}, y_{n-1})\}$ such that $\forall i \in \{0, ..., n-1\}, x_i \in \mathbb{R}^p$ and $y_i \in \mathcal{Y}$. Let us artificially create a set of unlabeled instances. We split the dataset into the subset $L = \{x_0, ..., x_{l-1}\}$ labeled instances and $U = \{x_l, ..., x_{n-1}\}$ the remaining instances for which the label is hidden. For the remainder of this work, let us assume that several random splits have been performed such that the labeled set corresponds to 20% of the whole dataset. Let us define $X = L \cup U$ the dataset that will be manipulated. Based on L, the set of triplets T as defined in Definition 4 can be computed. Let us recall the distance d to be the Euc-

[5] implemented in the Scikit-learn python package ([Pedregosa et al., 2011])

[6] Implemented in the pytonh package *metric-learn*

lidean distance on $\mathbb{R}^p$. The objective is to learn a function ϕ mapping $\mathbb{R}^p$ to a representation space $\mathbb{R}^q$ such that the Euclidean distance satisfies the constraint $d(\phi(x_i), \phi(x_j)) < d(\phi(x_i), \phi(x_k))$ for as many triplets $(x_i, x_j, x_k) \in T$ as possible. Let us learn the mapping function ϕ by minimizing the cost function $C(\phi|T)$ introduced in Equation 4.5.

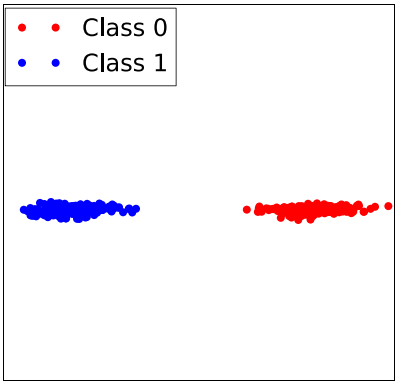
To handle complex non-linearly separable datasets, we aim at learning a neural network based mapping function ϕ. In order to minimize the cost function $C(\phi|T)$, let us implement a siamese neural network, having ϕ as core network, as introduced in Section 4.3.The siamese neural network used to learn ϕ is implemented with the Torch library ([Collobert et al., 2011]). The mapping function ϕ is a neural network and is consequently composed by a set of layers and activations function. For simplicity, let us assume that each neuron composing the neural network, except from the output layer, has the hyperbolic tangent $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ for activation function. The neural network ϕ is characterized by the set of weights composing its layers. The set of weights depends on the width of each layer. The number of layers and the width of each layer is chosen among a set of architecture by using a validation set. The function ϕ is learned by upgrading the weights parametrizing the neural network for each triplet of instances.

Once the mapping function ϕ is learned, let us analyse the representation space the data is projected in. To first have an intuition on the representation space in which the dataset is projected through ϕ, let us first analyse the two-dimensional PCA projection of the dataset in the representation space mapped by ϕ.

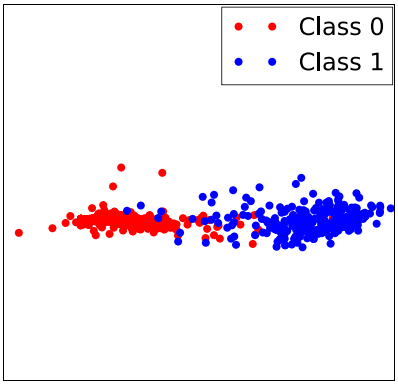
Figure 6.7 illustrates some of the projections of datasets in the representation space obtained through the *MDRL* algorithm. It can be observed that for some datasets, the algorithm clearly manages to separate the different classes, like for the *perturbedCircle3* dataset. For datasets that were originally somehow separated, like the *cancer* dataset, we can see that the instances from similar classes are grouped in more dense regions. Let us now consider datasets where classes are bit mixed, like the *vehicle* or the *digits* datasets. It can be observed that, even if in the representation space mapped by ϕ the different classes are not totally separated, classes are more dense and inter-classes boundaries are more clear through the PCA projection.

The proposed algorithm seems to be able to learn a mapping function projecting the data in a representation space which is more adapted to separate the different classes.

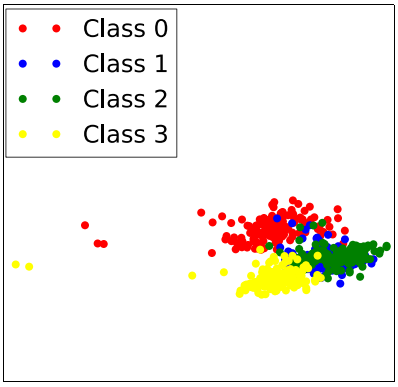
The first intuitions extracted from Figure 6.7 are based on visual observations. More than a visual analysis and confirmation of the capability of the algorithm to project the data into a more adapted representation space, let us more precisely analyse the representation space in which each data-



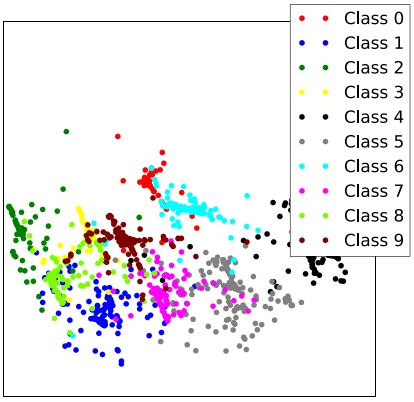
(a) 2D PCA of *perturbedCircle3* dataset



(b) 2D PCA of *cancer* dataset



(c) 2D PCA of *vehicle* dataset



(d) 2D PCA of *digits* dataset

Figure 6.7 – Two-dimensional PCA projections of the vectorial representation learned through the *MDRL* representation learning algorithm of some datasets

set is projected through the percentage of triplets satisfying their associated relative constraints. Let us compare the percentage of unsatisfied triplets constraints in the representation space of the different settings that will be manipulated for each dataset (Table 6.4). The number of possible triplets from a set of instances X increases with the size of the dataset. In order to practically evaluate the percentage of unsatisfied triplets constraints, let us sample a finite subset $T^* \subseteq T$ of the whole set of possible triplets. The number of triplets constituting T^* for each dataset is summarized in Table 6.3.

Dataset	Number of triplets
circle	1557
perturbedCircle1	1557
perturbedCircle2	1557
perturbedCircle3	1557
cancer	3606
ionosphere	495
vehicle	1970
digits	25972

Table 6.3 – Size of the set of triplets T^* for the different datasets

The percentage of non satisfied constraints is computed on the subset T^* of triplets. To neglect random samplings influence, the scores compiled in Table 6.4 are the mean of multiple iterations. Results compiled in Table 6.4 confirm the intuition based on the PCA projection of our datasets. It can be seen that globally, the representation space learned through our *MDRL* algorithm satisfies more constraints than the others settings. Let us focus on artificial datasets. A first observation that can be made is that our algorithm is able to satisfy most of the triplets based constraints for the *circle*, *perturbedCircle1* and *perturbedCircle3* datasets, while only half of the relative constraints are satisfied in the other settings. The percentages of unsatisfied constraints in the other settings for the *perturbedCircle2* are lower than for the other artificial datasets. An intuition that could explain the lower percentage for the *perturbedCircle2* dataset is that, due to the dataset creation and labeling processes, for half of the instances, the Euclidean distance on the last two features is representative of the task similarity and the Euclidean distance is globally more relevant than for the *circle* dataset. Simultaneously, it can be observed that although there are more unsatisfied constraints in the representation space learned by our algorithm for the *perturbedCircle2* dataset than for the other artificial datasets, the percentage of unsatisfied constraints for the *perturbedCircle2* in the *MDRL* setting is much lower than the percentage for the other settings.

Let us now consider the real world datasets. From Table 6.4, it can be seen that the *cancer* and the *digits* datasets are the most well separated real world datasets, regardless to the settings. Although the percentage of unsatisfied constraints in the *initial*, *feature selection* and *LMNN* settings are much lower than for the artificial datasets, it can highlighted that the *MDRL* setting allows us to have even less unsatisfied relative constraints.

In the worth cases, for the *ionosphere* and the *vehicle* datasets, the initial representation space satisfies more relative constraints than for the artificial

Dataset	MDRL	Initial	Feature selection	LMNN
circle	0.0	48.79 (± 1.98)	48.43 (± 1.43)	48.45 (± 1.9)
perturbedCircle1	0.0	50.63 (± 1.0)	50.31 (± 0.9)	50.52 (± 1.1)
perturbedCircle2	3.52 (± 2.9)	40.08 (± 2.0)	41.43 (± 2.2)	41.21 (± 2.0)
perturbedCircle3	0.23 (± 0.4)	49.62 (± 1.0)	48.86 (± 0.4)	49.45 (± 0.6)
cancer	4.37 (± 1.1)	7.16 (± 0.2)	7.26 (± 0.4)	8.22 (± 4.0)
ionosphere	16.36 (± 3.0)	39.19 (± 0.6)	36.75 (± 1.5)	35.89 (± 3.2)
vehicle	12.25 (± 2.4)	42.34 (± 1.5)	41.89 (± 1.0)	34.50 (± 1.8)
digits	1.48 (± 0.2)	12.16 (± 0.1)	11.30 (± 3.6)	11.41 (± 1.2)

Table 6.4 – Percentage of non satisfied relative constraints (± standard deviation) for the different datasets depending on the settings

datasets, except for the *perturbedCircle3* dataset that performs similar results. For both *ionosphere* and *vehicle* datasets, it can be observed that the *feature selection* and the *LMNN* settings gradually decrease the number of unsatisfied relative constraints, by proposing a better pair of vectorial representation and distance. Even if the percentage of unsatisfied triplets is higher for the two real world datasets with the *MDRL* setting than for the other real world datasets, the *MDRL* setting satisfies much more constraints than the other settings, with a decrease around 50% of the number of unsatisfied triplets compared to the *initial*, *feature selection* or *LMNN* settings. The higher percentage of unsatisfied constraints can be explained by the fact that due to the dataset size — less than 600 instances —, a small overlap between two different classes in the representation spaces proportionally impacts more triplets than the same overlap in a bigger dataset, like the *digits* dataset. Percentage of triplets satisfaction in such datasets is consequently highly influenced by blurry boundaries between classes.

From the described results, it appears that our algorithm allows us to project the data in a representation space where similarly labeled instances are globally closer from each other than from dissimilarly labeled instances.

6.5 Empirical evaluation of the theoretical assumptions

The theoretical analysis has shown that an optimal label propagation can be ensured if some assumptions are satisfied. The first assumption on which relies the theoretical analysis is related to the instances distribution in the initial vectorial representation space. The theoretical analysis also relies on the supposed capability of the representation learning algorithm to learn a vectorial representation of the data that minimizes the following cost func-

tion:

$$C(\phi|T) = \sum_{(x_i,x_j,x_k)\in T} max(0, \mu - [d(\phi(x_i), \phi(x_j) - d(\phi(x_i), \phi(x_k))])$$

where μ is the margin. The introduced assumptions are strong hypothesis. To evaluate the adequacy of the theoretical analysis to the real datasets, let us stress the two main assumptions of the theoretical analysis for the different datasets.

Let us first focus on the assumption that the vectorial representation is optimally learned. The mapping function ϕ is supposed to be learned such that all the relative constraints are satisfied at least from a margin μ. Let us split the set of labeled examples L into two sets, the training set, on which ϕ is learned and an evaluation set. Let us consider the value of the cost function for the two introduced sets for the different datasets (Table 6.5), while considering the percentage of unsatisfying constraints in the representation space obtained through the mapping function ϕ.

Dataset	constraints (%)	Training set	Evaluation set
circle	0.0	2.75e-5 (± 3.48e-5)	0
perturbedCircle1	0.0	1.01e-4 (± 1.4e-4)	0
perturbedCircle2	3.52 (± 2.9)	0.07 (± 0.14)	0.076 (± 0.15)
perturbedCircle3	0.23 (± 0.4)	0.02 (± 0.04)	0.027 (± 0.05)
cancer	4.37 (±1.1)	8.85e-4 (± 8.29e-4)	0
ionosphere	16.36 (±3.0)	4.46e-4 (± 3.42e-4)	0
vehicle	12.25 (±2.4)	1.3e-3 (± 5.65e-4)	0
digits	1.48 (± 0.2)	1.55e-3 (± 1.51e-3)	0

Table 6.5 – Cost function values (± standard deviation) obtained on the different sets of instances for the different datasets compared to the percentage of unsatisfied constraints with the *MDRL* algorithm

From Table 6.5, we can observe that the value of the cost function on the whole set of triplets computed on the training set if comparable to zero for most of the datasets, regardless to the considered random split, except for *perturbedCircle2* and *perturbedCircle3* datasets. Let us focus on *perturbedCircle2* and *perturnedCircle3* datasets. The value of the cost function for the evaluation set is, very close to the train set, non-zero. From previous experiments, the *perturbedCircle2* and *perturnedCircle3* datasets are the only two artificial datasets for which some constraints remain unsatisfied.

Let us now focus on the real world datasets. We can observe that the value of the cost function for the evaluation set is zero for the real world datasets and small for the training set. From previous experiments, it has

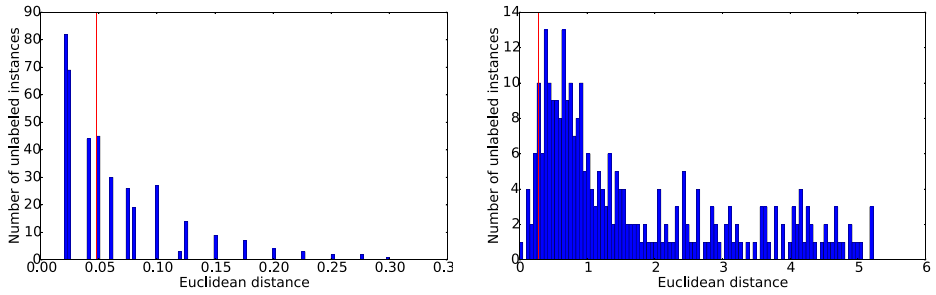
however been highlighted that part of the relative constraints are not satisfied, notably for the *ionosphere* and the *vehicle* datasets. It appears that most of the unsatisfied constraints must belong to the testing set.

For a triplet (x_i, x_j, x_k), the loss function as defined in Equation 4.3 can be non-zero even if the related constraint $d(\phi(x_i), \phi(x_j)) < d(\phi(x_i), \phi(x_k))$ is satisfied, due to the margin. The loss function is consequently non-zero whenever the margin μ is not satisfied. The percentage of unsatisfied constraints can not be related to the value of the cost function, as it appears in Table 6.5. On the opposite, a zero value of the cost function corresponds to the fact that the margin is achieved for the considered set of triplets, resulting in the satisfaction of their associated relative constraints.

The other strong assumption on which is based the theoretical analysis is related to the unlabeled instances distribution. Unlabeled instances are supposed to be close enough from at least one similarly labeled instance. More precisely, the algorithm assumes that the smallest distance between an unlabeled instance and a similarly labeled instance is smaller than a computable threshold η. η is defined as $\eta = \frac{\Delta^-_{\phi^m} - \Delta^+_{\phi^m}}{4\gamma(\phi)}$, where $\gamma(\phi)$ is a value depending on the parameters of ϕ (cf proof of Lemma 3 in Section 5.3). The term $\Delta^+_{\phi^m}$ (respectively $\Delta^-_{\phi^m}$) is the largest (respectively smallest) distance between training instances of the same class (respectively different classes) in the representation space mapped by ϕ. The threshold η consequently depends on the learned ϕ, on the specified distance d and on the set of labeled instances X_L. Its value varies for each dataset depending on the representation learning computation. The value of η is related to how well separated the different classes are in the representation space mapped by ϕ, depending on the complexity of the mapping function. By definition, η is computed if and only if the largest distance between similarly labeled training instances is smaller than the smallest distance between dissimilarly labeled training instances. For some datasets, the threshold η can be undefined, notably if there exists at least one pair of similarly labeled training instances that are more distant than one of the pair of dissimilarly labeled training instances. In our experiments, the threshold η can for example not be defined for some runs on the *vehicle* and the *digits* dataset. For the two datasets, the *MDRL* algorithm did not manage to separate the training instances depending on their classes.

Let us analyse how much unlabeled instances satisfy the assumption that they have a similarly labeled instances which is distant of at most η, considering the Euclidean distance in the initial representation space. Among the datasets and training/testing splits for which the threshold η is defined, let us for example focus on the *circle* and the *ionosphere* datasets. For the *circle* dataset, an average of 47.98% of the unlabeled instances, with a standard deviation of 32%, do not satisfy the assumption. Considering the *ionosphere*

dataset, the assumption is not satisfied for 98.69% of the unlabeled instances in average (±2.44). For part of unlabeled instances of both *ionosphere* and *circle* datasets, the minimal distance to a similarly labeled instance is consequently greater than the computed η. Let us consider the distribution of the distances between the unlabeled instances and their closest similarly labeled instances (Figure 6.8).



(a) *circle* dataset (b) *ionosphere* dataset

Figure 6.8 – Distribution of the minimal distance between each unlabeled instance and a similarly labeled instance, compared to the threshold η (red line)

In Figure 6.8, we can observe the distribution of the distance between each unlabeled instance and its closest similarly labeled instance in the initial representation space, for one of the training and testing splits. The red vertical line corresponds to the associated value of η. For the *circle* dataset, we can observe that most of the minimal distances related to unlabeled instances are close to the η threshold. More precisely, when most of them are below the threshold, it can be observed that fewer instances are concerned as the distance to the closest similarly labeled instance increase. A similar observation concerning the number of increasing distances can be made for the *ionosphere* dataset. For the *ionosphere* dataset, it can be highlighted that even if most of the pairwise distances are greater than η, many of them are located near to η. For both those datasets, the assumption on the instances distribution in the initial representation space is not satisfied.

From the introduced experiments, it seems that the two assumptions on which the theoretical analysis is based are not satisfied for some datasets. Let us empirically evaluate how the classification is performed, regardless of the theoretical assumptions being met or not.

6.6 Label propagation classification evaluation

The main goal of our work is to project the data in a representation space allowing us to perform an optimal graph-based classification. Let us assume that ϕ is learned such that it minimizes the cost $C(\phi|T)$ for a dataset X. Let us now evaluate the influence of the representation learning step on the classification solution. To evaluate the improvement due to our algorithm, let us compare the classification accuracy obtained with the *MBLR* setting to the other settings previously introduced.

The classification task is solved by applying the label propagation algorithm introduced in Section 3.3. To apply the label propagation algorithm, a graph representation of our dataset needs to be built. In each representation space, an ϵ-graph is built from the dataset X. The parameter ϵ is chosen on a validation set with a dichotomous approach. The label propagation algorithm introduced in Section 3.3 is then applied in order to classify unlabeled instances from U.

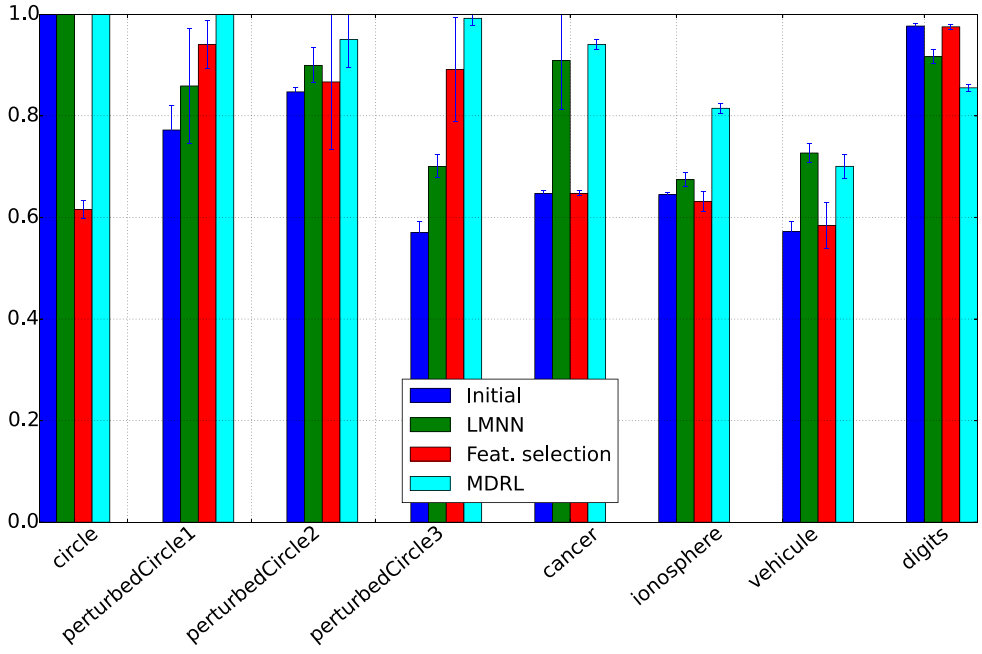


Figure 6.9 – Classification accuracy depending on the datasets and the settings

Classification results for the different datasets and (metric, representation space) settings are grouped in Figure 6.9.

Let us consider the artificial datasets. As the perturbations introduced

in the dataset increase, we can observe that the classification accuracy obtained with the different settings decreases. The highlighted behaviour is less pronounced for the *MDRL* setting than for the other.

As expected, the Euclidean distance in the *initial space* and *LMNN* setting perform ideally for the *circle* dataset, as the initial space is locally representative of the labeling similarity. We can also see that our *MDRL* algorithm achieves the same result. The poor performance of the label propagation algorithm applied on the *feature selection* setting, compared to the initial representation space, can be explained by the feature selection algorithm which kept features with a importance score equal or higher to a specific threshold. In our framework, the threshold is set to be the mean of all the scores. For the *circle* dataset, only one of the two features is kept if both features do not have exactly the same score, which is unlikely to happen due to the random nature of the representation learning algorithm.

In the *perturbedCircle1* dataset, the initial *circle* is blurred by uniform noise and, as expected, the Euclidean distance in the *initial space* and *LMNN* setting performances decrease compared to the *circle* dataset. We can observe that the *feature selection* setting gets better result in *perturbedCircle1* dataset than in *circle*, as it more easily extracts the two representatives features. When the *feature selection* setting is able to extract the circle-based feature, it performs similarly as the Euclidean distance did in the *circle* dataset. Finally, we can see that our *MDRL* algorithm manages to extract a good representation out of the initial space for the label propagation algorithm.

In the *perturbedCircle2* dataset, the initial *circle* features are blurred with features containing a structured noise. Results achieved in the *initial space*, *LMNN* and *feature selection* settings are similar to each others. Even if a structured noise is introduced, the Euclidean distance is locally pretty representative of the task similarity, allowing the label propagation algorithm to perform better classification in the *initial space* and the *LMNN* settings than for the *perturbedCircle1* dataset. Our *MDRL* algorithm still performs very good results on the *perturbedCircle2* dataset, learning a representation that separates well our dataset.

For the last artificial dataset, the label propagation algorithm in the *initial space* and the *LMNN* settings achieve poorer results than for other artificial datasets. The Euclidean distance not being representative of the task similarity, they are not able to extract the needed information. The *feature selection* setting performs similar results than for the previous datasets as the more representative features of the representation can easily be extracted by the feature selection algorithm, ignoring most of the introduced perturbations. On the other side, our *MDRL* algorithm was able to learn an appropriate representation space, allowing the label propagation to perform better results.

Let us now analyse the classification results obtained for on the real world

datasets.

Concerning the real dataset *cancer*, we can observe that the *LMNN* and our *MDRL* settings were able to modify the pair (representation, distance) such that the classification is quite well performed. Even if a linear transformation of the description space was sufficient to obtain pretty much good results, our non-linear representation learning algorithm was able to learn an efficient projection of the data. The other algorithm seems to face difficulties extracting the representative informations for classification.

For the *ionosphere* dataset, we can see that our *MDRL* algorithm performs better than the three others, each of them having similar results. It can be seen that the representation learned by our algorithm is much more adapted for the graph-based classification than that of the others algorithms. It can be explained by the initial distribution of the data, which seems to be blurred. Our non-linear representation learning is more adapted for separating the different classes.

Considering the *vehicule* dataset, we can see the interest of learning a representation space to obtain a better label propagation. Both *LMNN* and *MDRL* settings perform better results that the *initial space* or the *feature selection* settings. The initial blurred distribution of the dataset can explain the difficulty of the Euclidean distance to represent the task similarity. The non-linear projection seems more adapted to the dataset complexity.

A final observation can be made concerning the *digits* dataset, for which all settings performed well. It can be seen that, on the opposite to previous datasets, the best results are performed by the *initial space* and the *feature selection* settings. For the *digits* dataset, our algorithm, although performing pretty well, performs a lower accuracy than the *LMNN* setting. From previous experiments, it has been highlighted that the *MDRL* setting was the one minimizing the number of unsatisfied constraints. The described counter-intuitive result can be explained by the fact that even if less constraints are unsatisfied, the boundaries between some classes may be more blurry due to the projection. It has also be seen in Section 5.1 that satisfaction of the smoothness related constraints does not ensure a label propagation to optimally solve the task. Using only 20% of the dataset as a training set with much more classes than the other datasets may have been detrimental to learn the most adapted representation space.

Considering results compiled in Figure 6.9, it can be seen that a satisfying graph-based label propagation can be obtained by applying the *MDRL* algorithm, even if the theoretical assumptions are not met, as seen in Section 6.5. From the analysis of the empirical applicability of the theoretical analysis assumption and their impact on the label propagation algorithm accuracy, it seems that the assumptions on which is based the theoretical analysis are not always required to ensure an optimal classification. Relaxation of the initial conditions as required in the theoretical analysis may be

defined. The results obtained on the several experiments show that a more general theoretical analysis can be performed in order to define a relationship between the *MDRL* representation space learning and the label propagation accuracy.

6.7 Do we need a graph?

In this work, we focused on solving a classification task through a label propagation algorithm applied on an ϵ-graph. The use of an ϵ-graph is justified in the theoretical analysis, as we have demonstrated to have an optimal graph for label propagation. The defined ϵ-graph is however not the unique possible graph representation that can be obtained from a dataset. Similarly, other classification algorithms that do not reason from a graph representation of the data can be used to solve the targeted task. In the present section, we discuss about the need for a ϵ-graph representation of the dataset to solve the task based on the vectorial representation of the data learned through the *MDRL*. We more generally evaluate the need of a graph-based classification algorithm.

The question of the generalization of the theoretical work to k-nn graph representation can be asked. Let us focus on the empirical analysis of the application of the label propagation algorithm on a symmetric k-nn graph representation of each dataset. Similarly to the ϵ tuning, the optimal k is chosen through a validation set. The classification accuracy obtained based on a k-nn graph (Figure 6.10) is compared through the different settings.

Let us first briefly focus on comparing the label propagation classification that can be obtained on a k-nn graph built in the different settings. Similarly to the ϵ-graph, it can be highlighted that the k-nn simplification of the graph representation of each dataset built in the representation space proposed by the *MDRL* algorithm allows us to obtain a satisfying classification through the label propagation algorithm. More precisely, we can observe that our MDRL setting is among the settings performing at best the classification on most of the dataset, except for the *digit* dataset. For other settings, results are globally similar to the results obtained by applying an ϵ simplification on the graph representations of the datasets (Figure 6.9).

Let us now more precisely focus on comparing the label propagation accuracy obtained on an ϵ or a k-nn graph in the MDRL setting.

It can be seen in Table 6.6 that the results obtained on the k-nn graph representation of the datasets are very similar to the results obtained with the ϵ-simplification. This can be explained by the fact that depending on the parameters tuning, ϵ simplification and k-nn simplification can lead to pretty similar neighbourhood for each instance of the graph. With very similar

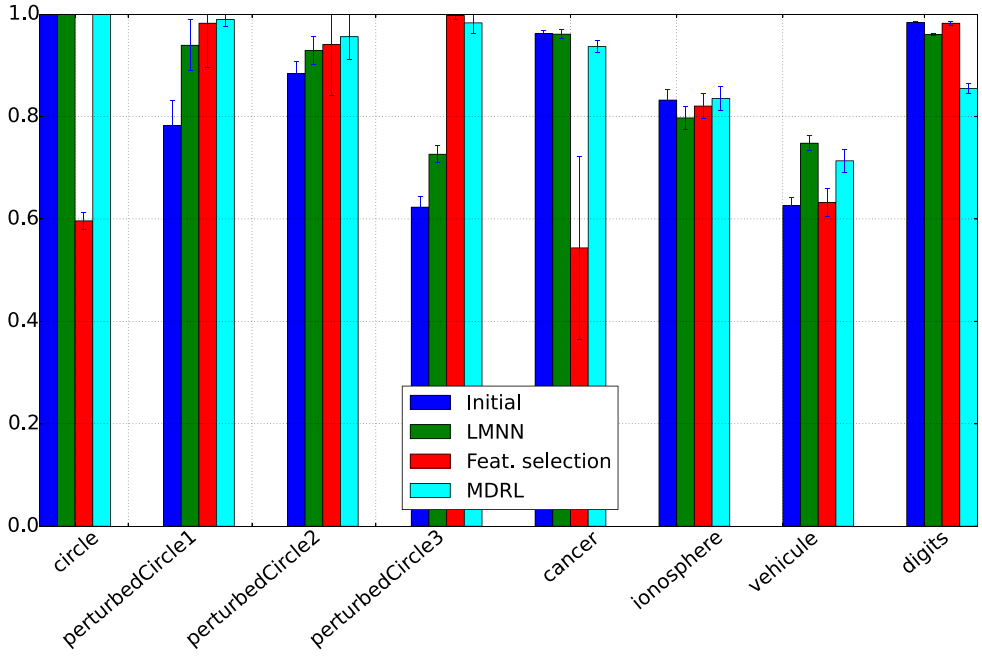


Figure 6.10 – Classification accuracy on a symmetric k-nn graph depending on the datasets and the settings

Dataset	ϵ-graph	k-nn graph
circle	1.0 ($\pm$ 0.0)	1.0 ($\pm$ 0.0)
perturbedCircle1	1.0 ($\pm$ 0.0)	0.99 ($\pm$ 0.01)
perturbedCircle2	0.95 ($\pm$ 0.05)	0.96 ($\pm$ 0.04)
perturbedCircle3	0.98 ($\pm$ 0.02)	0.98 ($\pm$ 0.02)
cancer	0.94 ($\pm$ 0.01)	0.94 ($\pm$ 0.01)
ionosphere	0.81 ($\pm$ 0.01)	0.83 ($\pm$ 0.02)
vehicle	0.70 ($\pm$ 0.02)	0.71 ($\pm$ 0.02)
digits	0.85 ($\pm$ 0.01)	0.85 ($\pm$ 0.01)

Table 6.6 – Label propagation accuracy ($\pm$ standard deviation) depending on the datasets and the graph construction methods for the MDRL setting

graph structure, the label propagation behaviour is similar. More precisely, if we are able to define the parameter k and ϵ allowing each simplification methods to be as close as possible to the optimal graph, the highlighted graph structure would be similar and so would the label propagation classification results.

Based on a symmetric k-nn graph, label propagation algorithm is able to

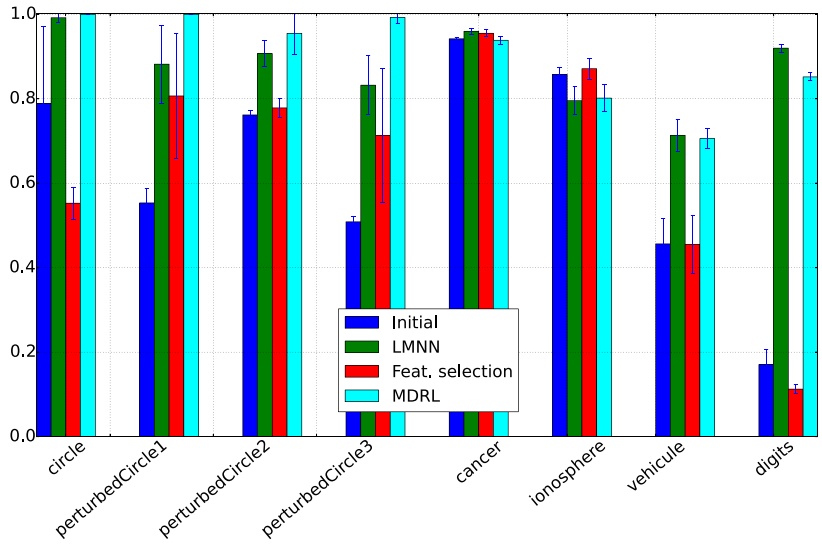
reasonably classify unlabeled instances based on few labeled examples. From the experimental results obtained with the k-nn simplification, it seems that neither the ϵ simplification or the k-nn simplification can be preferred to apply the label propagation algorithm. The previous observation allows us to work on generalization of the theoretical analysis to k-nn simplification.

This work focused on the resolution of the classification task through a graph-based label propagation algorithm. The representation space of the dataset is consequently learned such that similarly instances are closer from each other than to dissimilarly labeled instances. The property of the representation space, related to the smoothness assumption, has already been highlighted. In the representation space learned through the *MDRL* algorithm, the instances are distributed to be, at least non-linearly, separable. Some other semi-supervised classification algorithm take advantage of the classes separability assumption to solve the task. The representation space learned to satisfy the separability property may consequently by advantageous for other classical classification algorithms. Let us compare the application of some other classification algorithms for the different settings (Figure 6.11). More precisely, let focus on two non graph-based classification algorithms[7], a multi-class SVM-based classifier (Figure 6.11a) and a tree-based classifier (Figure 6.11b). In a "one-against-one" approach, the SVM-based classifier trains multiple binary classifiers ([Guyon et al., 1993]). The tree-based classifier ([Geurts et al., 2006], cf Section 2.4) aims at classifying the dataset by splitting the data depending on the value of the different variables. A set of rules can be extracted from the trees in order to predict the label of unseen instances.

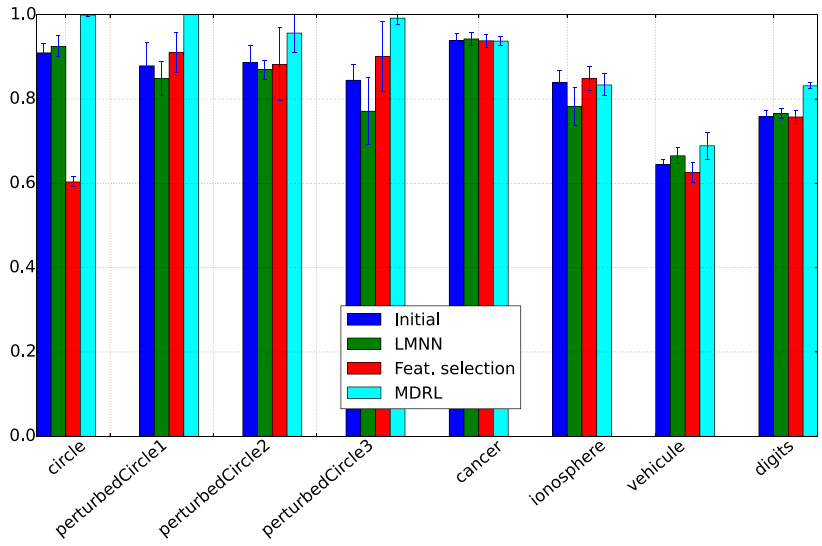
From Figure 6.11, it can be seen that results obtained with the tree-based classifier are quite homogeneous. The results obtained for each dataset are not significantly different regarding the setting on which the classifier is learned and applied. It can be highlighted that the *MDRL* setting is among the best settings considering the accuracy. The observation about settings homogeneity is not verified for the SVM based classifier. For the SVM based algorithm, the classification accuracy obtained for each dataset is more dependent on the selected setting. It can be seen that the SVM based classifier best results are globally obtained in the *LMNN* or the *MDRL* settings, except for the *ionosphere* dataset.

Let us now compare the resolution of the classification task in the MDRL setting through the application of the two non graph-based algorithms to the label propagation algorithm applied on an ϵ-graph (Table 6.7).

[7]Both algorithms are implemented in the Python *Scikit learn* module ([Pedregosa et al., 2011])



(a) SVM based classifier



(b) Tree based classifier

Figure 6.11 – Classification accuracy depending on the datasets and the settings obtained with an SVM based classifier and a tree based classifier

In Table 6.7, it can be seen that the SVM based classifier and the tree based classifier performances are very similar to the results obtained with the label propagation algorithm in the *MDRL* setting, for all the datasets. The representation space learned by the introduced *MDRL* algorithm allows

Dataset	Label propagation	SVM classifier	Tree classifier
circle	1.0 (± 0.0)	1.0 (± 0.0)	0.99 (± 0.003)
perturbedCircle1	1.0 (± 0.0)	1.0 (± 0.0)	1.0 (± 0.0)
perturbedCircle2	0.95 (± 0.05)	0.95 (± 0.05)	0.96 (± 0.04)
perturbedCircle3	0.98 (± 0.02)	0.99 (± 0.01)	0.99 (± 0.01)
cancer	0.94 (± 0.01)	0.94 (± 0.01)	0.94 (± 0.01)
ionosphere	0.81 (± 0.01)	0.80 (± 0.03)	0.83 (± 0.03)
vehicle	0.70 (± 0.02)	0.71 (± 0.02)	0.69 (± 0.03)
digits	0.85 (± 0.01)	0.85 (± 0.01)	0.83 (± 0.01)

Table 6.7 – Classification accuracy (± standard deviation) depending on the datasets and the classification algorithms in the *MDRL* setting

us to efficiently apply different classification algorithms. Several questions emerge from the brief analysis of the obtained results. The introduced results first allow us to question the generalization of the theoretical analysis to other graph construction methods and classification algorithms. Among the questions that can arise, the need for a graph representation to optimally solve the targeted task in the representation space learned through the *MDRL* algorithm is questionable.

Chapter 7

Conclusion & future work

7.1 Conclusion

The objective of the present work was to answer the question of the interest of a task driven representation learning for the targeted task resolution. It has indeed been highlighted that the resolution of a supervised and semi-supervised *Machine learning* task is highly influenced by the adequacy of the representation space and the distance used. In the literature, numerous approaches have been developed in order to tune the representation space and the distance together, such that the application of the distance in the representation space is representative of the task similarity. The developed methods can be unsupervised but many of them are driven by a set of constraints, which can be related to the task.

This work more particularly focused on the classification task, solved by a graph-based algorithm. To apply the label propagation algorithm introduced in Section 3.3, a graph representation of the data needs to be built. Graph-based algorithms usually assume that the graph is a homophilic structure. The objective is thus to build a graph such that it satisfies the smoothness assumption. We proposed a representation learning algorithm such that an ϵ-graph or a k-nn graph built in the representation space proposed by the algorithm can satisfy the smoothness assumption.

A first theoretical analysis has been performed in order to have guarantees on the classification that needs to be performed on the obtained ϵ-graph. Several empirical evaluations have been performed to evaluate the interest of such an approach. Although the theoretical analysis and empirical evaluations have shown the interest of such method for classification improvement, it has also been highlighted that the theoretical analysis can be relaxed, extended. In the following, we provide some insights on directions to explore that might achieve more general guarantees.

7.2 Open questions

Theoretical guarantees and several experiments have shown the interest of the introduced *MDRL* algorithm to outperform the initial representation space for graph-based label propagation algorithm. Several questions and possible extensions however emerged from the experiments that have been performed.

A first drawback of our theoretical analysis is that the analysis is built such that we define the conditions needed in order to have a specific result. No guarantees are thus available if one of the initial conditions is not satisfied, which may be common as our assumptions are pretty strong, as seen in Section 6.5. An improvement of the theoretical analysis would be to bound the classification error that can be expected depending on the representation learning cost that is achieved. Let us introduce some leads for new bounds and theoretical guarantees.

Bounding the classification error depending on the representation learning algorithm can be achieved in two main steps, which are not trivial. At first, properties of the graph representation that can be obtained must be linked to the representation learning phase. The expected classification error can then be bounded depending on the graph structure. The two frameworks have been briefly initiated.

Let us first focus on linking the graph-based classification algorithm behaviour and results to the graph structure. As previously discussed in Section 3.4, authors of [Maier, Markus et al., 2013] studied the influence of graph construction methods on a measure of clustering quality. In their work, they link the *NCut* and the *Cheeger Cut* measures, which are two normalized variants of the cut measure, to the structure of a graph and some of its properties, depending on its construction method. The introduced paper has numerous interesting points. The main advantage their framework has over ours is that their approach compares different graph construction methods and extract some graph properties depending on the construction method. Although the work presented in [Maier, Markus et al., 2013] is close to our objective, it is not directly applicable to our context. They indeed consider some clustering related quantities when we are concerned by classification problems. The set of assumptions that are supposed to be satisfied in their work is another element which does not match our concerns. They indeed assume some strong hypothesis on the instances distribution and separability. Despite the introduced drawbacks, there are some interesting theoretical considerations on which could be based a new theoretical analysis to extend the one proposed in the present thesis.

Instead of directly linking a graph construction method to a *Machine learning* related quantity, some works propose to link the structure of a graph

to classification related quantity. Authors of [Guillory and Bilmes, 2009] thus propose different bounds for graph-based classification depending on the graph structure, considered as fixed, and the classification algorithm. Considering a fixed graph representation of a dataset, they indeed aim at choosing the set of labeled instances allowing to perform the most optimal classification, while the set of labeled instances is fixed in our framework. To choose the best set of labeled instances, they express a first general bound for the classification error. They bound the classification error depending on the connectivity of the unlabeled instances to the remainder of the graph, the labeling and the structure of the graph. The more the labels are smooth over the graph and the fewer isolated unlabeled instances there are, the smaller the classification error is. More precisely, they assume that each unlabeled instance is connected to the set of labeled instances from at least a specific threshold, regardless of their labels. The more relaxed distribution assumption of the data and the bounds that are defined in [Guillory and Bilmes, 2009] are interesting for a new work on a generalization of our theoretical analysis. Some questions can however be highlighted concerning the usefulness and applicability of the defined bound. Another drawback of the work from [Guillory and Bilmes, 2009] is that the authors considered the graph fixed but in our framework, we do not have any guarantees about the graph structure and thus the final bound we can obtain.

Let us assume that the classification error can be bounded depending on the graph structure. A second main step is to define the graph structure and properties depending on the representation learning step. More precisely, we first seek at bounding the objective function C depending on the initial space. Based on the value of C, the objective would then be to define the structure and properties of the graphs that can be obtained.

An approach in order to solve the introduced problems could be based on works found in [Clémençon et al., 2008]. In the introduced paper, authors aim at defining a statistical framework for ranking problems. In their work, they define theoretical bounds and results for U-statistics. As they are able to express the ranking problems as U-statistics, they define theoretical bounds on the empirical risk of ranking problems. A guideline in order to link the graph structure to the representation learning step and to base a more general theoretical analysis can consequently pass through reformulating part of our problem as U-statistics. This way, generalization bound for the representation learning algorithm may be defined.

Due to other drawbacks of our theoretical analysis, some relaxations and extensions of the theoretical work introduced in the present thesis are possible. The theoretical analysis that we performed is based on two strong assumptions on the initial data distribution and on the optimally learned mapping function. From experiments, it indeed appears that for some data-

sets, a nearly optimal classification can be obtained in our working context although the theoretical assumptions are not satisfied. Some relaxation of the strong introduced assumptions may thus be possible and the theoretical analysis may be extended.

Another point that has been highlighted by the experiments concerning our theoretical analysis is related to the inherent methods involved in our working context. The theoretical analysis has been performed for a specific graph construction method — the ϵ simplification — and a set classification algorithm — the label propagation algorithm introduced by [Zhu and Ghahramani, 2002] —. Even if the assumptions of the theoretical guarantees can be relaxed, the present thesis did not provide any theoretical elements for graphs obtained with other graph construction methods or for other classification algorithms, graph-based or not. Performed experiments has shown that a k-nn simplification of the graph representation of datasets allows us to obtain similar classification results than results obtained with the ϵ simplification. We also observed the interest of such a representation learning algorithm for classification through some other classical classification algorithms. Similar classification results are obtained in the vectorial representation learning through the *MDRL* algorithm regardless of the simplification method applied during the graph construction process or of the classification algorithm used to solve the task. An extension of the theoretical guarantees for other graph construction methods and classification algorithms could be considered.

A question arises from the fact that similar results are obtained regardless the classification algorithm used to solve the task is graph-based or not. Graph-representations of a dataset allow graph-based classification algorithms to relax the need for a representation space where the different classes are clearly separated in dense clusters. For datasets where similarly labeled instances are not grouped in a unique cluster but in smaller dense and homogeneous clusters which can be closer from clusters from other classes than from similarly labeled clusters, a graph-representation of a dataset allows to extract the local structure useful to solve the targeted task.

From the several experiments performed in Chapter 6 on the introduced datasets, the representation space learned through the *MDRL* algorithm seems to separate the different classes in well separated unique clusters. In a representation space where the different classes are separated in dense clusters, graph-representation of the dataset and graph-based algorithms are not useful to solve the task. Classification algorithms that do not reason from a graph-representation of the dataset performing similar results that graph-based algorithms, it empirically appears that a graph representation of the data is not required to take advantage of the learned vectorial representation.

The *MDRL* algorithm aims at mapping the data in a representation space where similarly labeled instances all lie in the same dense cluster based on relative constraints defined on the whole set of instances. It has already been highlighted that graph-representations allow to relax the assumptions on how separated the instances from different classes are. Instances are consequently only required to be closer from a subset of similarly labeled instances. Another work direction based on the theoretical work presented in the thesis could be to train the *MDRL* algorithm with triplets of instances sampled such that only the local relative constraints aim at being satisfied. More precisely, triplets of instances (x_i, x_j, x_k) could be sampled such that x_i and x_j are closest than a set threshold. A mapping function should be more easily learned with such a variant of the *MDRL* algorithm as the representation space is less constrained. A graph-based classification algorithm could take advantage of the structure highlighted by the graph-representation built in the less constrained representation space to solve the task and may then outperform classical non graph-based classification algorithms.

7.3 Clic and Walk case study

The present work was initially lead by the needs of the *Clic and Walk* company. *Clic and Walk* is a start-up which proposes to perform market surveys and to collect customers opinions for other companies. Through a smartphone application, users, i.e. the customers, received a mission they have to completed in return for a remuneration. A mission is usually composed by a set of questions and a set of pictures they have to take. Due to the limited number of answers they can pay and the limited time available to give results back to the companies, the objective of *Clic and Walk* is to propose missions first and foremost to the most appropriate users. The introduced need can be transposed as a classification task, where users are associated with a binary label, appropriate or not.

The empirical intuition is that all the missions are not similarly fulfilled due to their inner characteristics. Some missions requiring more commitments from the user to be completed, some kind of missions are prone to be less easily answered. The various missions are consequently grouped together depending on their difficulties, for example the place they are required to be performed. The objective is then to associate with each user a label, for each type of mission.

Another objective the company has is to understand the users behaviour and characteristics. There is thus a need to structure the set of users and to analyse the obtained structure. An efficient approach to solve the latest issue is to build a graph representation of the set of users and then to analyse the structure of the obtained graph.

As the two described tasks are somehow related, we aim at solving them

conjointly. The objective is consequently to build a graph from the set of users. The obtained graph can be used for a graph-based classification, for example the label propagation algorithm, and for a clustering analysis. Due to the real world context and the available information about the users, the Euclidean distance, that can be used to build the graph, is poorly representative of the task in the initial representation space of the users. Inadequacy of classical distances for real world problems led our work.

During the algorithm definition, the theoretical analysis of the algorithm and the implementation in the company context, a first protocol has been performed to answer the recommendation issue. The classification step was performed with a decision tree based classifier from Scikit. A mission was then proposed to the limited set of users positively labeled. After a few days, the access of the mission was finally granted to all the users. The developed protocol was evaluated by comparing some measures like the percentage of persons accepting the mission, failing it or doing it right to the previous mission, without selection. The quality of the answers sent by the users was also compared. The preliminary results were encouraging and showed an improvement in the quality of the answers as well as the number of answers.

Bibliography

[Belkin et al., 2004] Belkin, M., Matveeva, I., and Niyogi, P. (2004). Regularization and semi-supervised learning on large graphs. In *In COLT*, pages 624–638. Springer.

[Bellet et al., 2013] Bellet, A., Habrard, A., and Sebban, M. (2013). A Survey on Metric Learning for Feature Vectors and Structured Data. *ArXiv e-prints*.

[Bengio et al., 2013] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

[Blum and Chawla, 2001] Blum, A. and Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 19–26, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA. ACM.

[Bottou, 2012] Bottou, L. (2012). *Stochastic Gradient Descent Tricks*, pages 421–436. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Bromley et al., 1994] Bromley, J., Guyon, I., Lecun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a "siamese" time delay neural network. In *In NIPS Proc.*

[Burges, 1998] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

[Cestnik et al., 1987] Cestnik, G., Konenenko, I., and Bratko, I. (1987). Assistant-86: A knowledge-elicitation tool for sophisticated users. In Bratko, I. and Lavrac, N., editors, *Progress in Machine Learning*, pages 31–45. Sigma Press, Ljubljana.

[Chapelle et al., 2006] Chapelle, O., Schlkopf, B., and Zien, A. (2006). *Semi-Supervised Learning*. The MIT Press, 1st edition.

[Chopra et al., 2005] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 539–546, Washington, DC, USA. IEEE Computer Society.

[Clémençon et al., 2008] Clémençon, S., Lugosi, G., and Vayatis, N. (2008). Ranking and Empirical Minimization of U-statistics. *Annals of Statistics*, 36(2):844–874.

[Collobert et al., 2011] Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011). Torch7: A Matlab-like Environment for Machine Learning. In *BigLearn NIPS Workshop*.

[de Sa, 1994] de Sa, V. R. (1994). Learning classification with unlabeled data. *Advances in neural information processing systems*, pages 112–112.

[de Sousa et al., 2013] de Sousa, C. A. R., Rezende, S. O., and Batista, G. E. A. P. A. (2013). Influence of graph construction on semi-supervised learning. In Blockeel, H., Kersting, K., Nijssen, S., and Zelezný, F., editors, *ECML/PKDD (3)*, volume 8190 of *Lecture Notes in Computer Science*, pages 160–175. Springer.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.

[Fischer and Igel, 2012] Fischer, A. and Igel, C. (2012). *An Introduction to Restricted Boltzmann Machines*, pages 14–36. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Fisher, 1938] Fisher, R. A. (1938). The statistical utilization of multiple measurements. *Annals of Eugenics*, 8(4):376–386.

[Frome et al., 2007] Frome, A., Singer, Y., Sha, F., and Malik, J. (2007). Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*, pages 1–8.

[Fukunaga, 1990] Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition (2Nd Ed.)*. Academic Press Professional, Inc., San Diego, CA, USA.

[Fürnkranz, 1999] Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54.

[Geurts et al., 2006] Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Mach. Learn.*, 63(1):3–42.

[Grover and Leskovec, 2016] Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653.

[Guillory and Bilmes, 2009] Guillory, A. and Bilmes, J. A. (2009). Label selection on graphs. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 691–699. Curran Associates, Inc.

[Guyon et al., 1993] Guyon, I., Boser, B. E., and Vapnik, V. (1993). Automatic capacity tuning of very large vc-dimension classifiers. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 147–155, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

[Hoffer and Ailon, 2014] Hoffer, E. and Ailon, N. (2014). Deep metric learning using triplet network. *CoRR*, abs/1412.6622.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366.

[J. Weston, 2008] J. Weston, F. Rattle, R. C. (2008). Deep learning via semi-supervised embedding. In *International Conference on Machine Learning*.

[Joachims, 1999] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 200–209, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Kedem et al., 2012] Kedem, D., Tyree, S., Weinberger, K., Sha, F., and Lanckriet, G. (2012). Non-linear metric learning. In Bartlett, P., Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 2582–2590.

[Kotsiantis, 2007] Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. *Informatica (Slovenia)*, 31(3):249–268.

[Kulis, 2012] Kulis, B. (2012). Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364.

[Le Cun, 1986] Le Cun, Y. (1986). *Learning Process in an Asymmetric Threshold Network*, pages 233–240. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Lecun, 1985] Lecun, Y. (1985). *Une procedure d'apprentissage pour reseau a seuil asymmetrique (A learning scheme for asymmetric threshold networks)*, pages 599–604.

[Maier, Markus et al., 2013] Maier, Markus, von Luxburg, Ulrike, and Hein, Matthias (2013). How the result of graph clustering methods depends on the construction of the graph. *ESAIM: PS*, 17:370–418.

[McPherson et al., 2001] McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444.

[Murthy, 1998] Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Perozzi et al., 2014] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652.

[Ramanan and Baker, 2011] Ramanan, D. and Baker, S. (2011). Local distance functions: A taxonomy, new algorithms, and an evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):794–806.

[Rifai et al., 2011] Rifai, S., Dauphin, Y., Vincent, P., Bengio, Y., and Muller, X. (2011). The manifold tangent classifier. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 2294–2302.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386.

[Rosenblatt, 1962] Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington. it Early work on what would now be referred to as a "connectionist" model.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA.

[Schölkopf et al., 1998] Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319.

[Scholkopf and Smola, 2001] Scholkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

[Weinberger and Saul, 2009] Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research (JMLR)*, 10:207–244.

[Yang, 2006] Yang, L. (2006). Distance metric learning: A comprehensive survey.

[Yarowsky, 1995] Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Zhang, 2000] Zhang, G. P. (2000). Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462.

[Zhou et al., 2004] Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 321–328. MIT Press.

[Zhu, 2005] Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

[Zhu and Ghahramani, 2002] Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University.

[Zhu et al., 2003] Zhu, X., Ghahramani, Z., and Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 912–919.

Task driven representation learning

Apprentissage de représentation dirigé par la tâche

Pauline Wauquier

Le doctorat durant lequel les travaux présentés dans ce document ont été menés s'est déroulé en collaboration avec la start-up Clic and Walk *(convention CIFRE). Cette entreprise effectue des études de marchés et collecte les avis des clients pour différentes entreprises. A la demande des entreprises mandatrices, des missions, composées par un ensemble de questions et de photos, sont envoyées aux utilisateurs de l'application* Clic and Walk*, qui perçoivent une rémunération en échange de leurs réponses. Différentes contraintes de temps et de coûts nécessitent de mettre en place une meilleure stratégie de distribution des missions aux différents utilisateurs.*

Au cours de ce doctorat, j'ai pu effectuer des recherches théoriques dans le but de mettre en place une recommandation des utilisateurs. Ce document présente les travaux théoriques qui ont été menés durant le doctorat sur l'apprentissage de représentation dirigé par une tâche.

De nombreux problèmes peuvent être résolus en prédisant une information à partir d'un ensemble de données. Afin d'illustrer nos propos, intéressons nous à un site de vente en ligne et quelques problèmes marketing pouvant lui être associés. Une fois inscrit, un utilisateur du site peut acheter et noter les différents produits qui sont disponibles. Les employés du site marchand peuvent accéder au profil des utilisateur, les notes que ces derniers ont donnés aux différents produits et leurs achats. Afin d'augmenter le nombre de ventes, les employés souhaitent inciter les clients à acheter plus de produits. Ils doivent donc pouvoir proposer à chaque client des produits susceptibles de l'intéresser. L'intérêt d'un utilisateur pour un produit spécifique doit donc être prédit à partir des informations disponibles, notamment le profil de l'utilisateur et les caractéristiques du produit. Une autre amélioration marketing serait de segmenter l'ensemble des utilisateurs dans le cadre d'une segmentation de marché, en se basant sur le profil des utilisateurs ou leurs historiques d'achats. Le site de commerce en ligne pourrait également souhaiter établir un budget prévisionnel. La somme qu'un client est susceptible de dépenser dans un intervalle de temps donné sur le site doit donc être prédite, en fonction des informations disponibles à propos du client.

Parmi les différentes tâches de prédiction qui peuvent être extraites des problèmes rencontrés en contexte réel, citons:

- **Classification**: une étiquette doit être associée à chaque entité (l'intérêt d'un client pour un objet par exemple)
- **Régression**: une valeur continue doit être associée à chaque entité (la somme qu'un client est susceptible de dépenser dans un intervalle de temps donné par exemple)
- **Clustering**: l'objectif est de définir une partition sur l'ensemble des entités (segmentation de marché par exemple)

Les problèmes de prédiction précédemment introduits peuvent être résolus en demandant à des experts du domaine d'analyser les données collectées afin d'extraire un schéma, de définir des règles ou d'extraire l'information voulue. Quelqu'un peut ainsi être employé par le site de commerce en ligne afin de créer des associations entre les différents produits et clients. Le site marchand traitant des ensembles toujours croissant de produits et de clients, créer chaque association à la main est chronophage et coûteux. Une meilleure solution pour le site est de pouvoir prédire automatiquement l'intérêt qu'un client peut avoir pour un produit spécifique en se basant sur les informations disponibles concernant l'utilisateur et le produit. De même, pour le problème de prédiction des dépenses des clients, un employé doit être assigné à l'analyse des dépenses des années précédentes de chaque clients ainsi que de leurs informations de profils. Plutôt que de résoudre manuellement cette tâche, le site marchand pourrait chercher à automatiquement prédire la somme qu'un client est susceptible de dépenser durant l'année courante en se basant sur son profil, son historique d'achats ainsi que sur l'ensemble des informations concernant les autres clients.

La définition manuelle d'un ensemble de règles et la recherche manuelle de schémas ne sont pas seulement coûteuses en temps et en argent. Les tâches ciblées peuvent aussi être trop complexes pour être résolues par un humain. Certains problèmes peuvent être trop compliqués, ou les données disponibles trop nombreuses, pour être résolus de manière optimale à la main. Résoudre manuellement un problème de segmentation de marché nécessite qu'un employé explore et analyse le comportement et les informations d'un ensemble de clients pour extraire des groupes cohérents. A cause du nombre de clients et des variables associées à chacun des clients, la segmentation de marché est un problème qui peut rapidement devenir trop complexe pour être résolu à la main.

Les algorithmes d'*apprentissage automatique* proposent de nombreuses approches répondant aux problématiques précédemment abordées. Les méthodes d'*apprentissage automatique* cherchent à résoudre de manière optimale les problèmes issus de contextes réels. Elles sont développées pour résoudre automatique une tâche en se basant sur un jeu de données, sans nécessiter l'implication d'aucun expert durant l'étape d'apprentissage.

Un jeu de données est un ensemble d'information à partir duquel les algorithmes d'*apprentissage automatique* traitent la tâche. Les jeux de données sont communément composés d'un ensemble d'entités à partir desquels un algorithme apprend ou pour lesquels il cherche à résoudre la tâche. Des informations supplémentaires sont parfois disponibles pendant le processus de création du jeu de données. L'étiquette d'une instance, associée à la tâche ciblée, peut ainsi être disponible pendant l'étape de création du jeu de données ou a pu être proposée par un expert. L'étiquette devant être prédite

dans le cadre de la tâche ciblée peut donc être disponible pour certaines entités. Le site marchand a par exemple accès à l'ensemble des produits que les clients ont acheté les années précédentes et peut donc facilement calculer le montant dépensé par les différents clients. Pour la recommandation de produits, l'intérêt des clients peut être déduits pour l'ensemble des produits ayant précédemment été notés ou achetés, à partir des historiques d'achats.

Les algorithmes d'*apprentissage automatique* peuvent exploiter ces informations supplémentaires. Selon l'utilisation ou non de cet information complémentaire, les algorithmes d'*apprentissage automatique* peuvent être regroupés en différentes catégories, parmi lesquelles nous pouvons citer les suivantes:

- **apprentissage supervisé**: une étiquette est disponible pour chaque entité du jeu données étant disponible durant l'entraînement. Les algorithmes supervisés cherchent à apprendre un modèle pouvant prédire une étiquette pour toute entité issue de la même distribution.
- **apprentissage semi-supervisé**: les jeux de données sur lesquels se basent les algorithmes semi-supervisés sont composés d'un ensemble d'entités étiquetées et d'un ensemble d'entités non-étiquetées. Un algorithme semi-supervisé exploite l'ensemble des entités disponibles, étiquetées ou non, pour prédire l'étiquette de toute entité issue de la même distribution.
- **apprentissage non-supervisé**: les algorithmes non-supervisés sont entraînés à partir de données non-étiquetées. Ils apprennent un modèle à partir de la représentation vectorielle de l'ensemble du jeu de données sans avoir accès à aucune étiquette. La tâche est résolue en se basant uniquement sur la distribution et la représentation vectorielle des données.

Dans ces travaux, nous nous concentrons sur la tâche de classification. Pour la suite, définissons les notations associées à la tâche de classification. Soit $\mathcal{X}$ un espace vectoriel et soit $\mathcal{Y}$ l'ensemble d'étiquettes discrètes $\{0, ..., c-1\}$. $\mathcal{P}$ est la distribution des données dans $\mathcal{X} \times \mathcal{Y}$. Soit $\{(x_0, y_0), ..., (x_{l-1}, y_{l-1})\}$ (respectivement $\{x_l, ..., x_{l+u-1}\}$) un ensemble d'exemples étiquetés (respectivement un ensemble d'entités non étiquetées), tels que $x_i \in \mathcal{X}$ et $y_i \in \mathcal{Y}$. On suppose que les exemples de chaque ensemble sont des variables indépendantes et identiquement distribuées. Les ensembles L et U sont définis tels que $L = \{x_0, ...x_{l-1}\}$ et $U = \{x_l, ..., x_{l+u-1}\}$. Un jeu de données X est défini comme étant l'union $X = L \cup U$. On suppose que les exemples dans X sont tirés selon la distribution $\mathcal{P}$. Nous définissons finalement $\mathbf{y_L} = [y_0, ..., y_{l-1}]$ le vecteur d'étiquettes associé aux entités composant L et $\mathbf{y_U} = [y_l, ..., y_{l+u-1}]$ le vecteur d'étiquettes que l'on cherche à prédire

pour les entités composant U. Le vecteur $\mathbf{y}$, issu de la concaténation de $\mathbf{y_L}$ et $\mathbf{y_U}$, et le vecteur d'étiquettes associé au jeu de données X.

Les algorithmes supervisé cherchent à retrouver la distribution des données et de leurs étiquettes sous-jacent au jeu de données, afin de prédire l'étiquette d'entités issue de la même distribution. Les algorithmes de classification supervisé apprennent un modèle associant une étiquette dans $\mathcal{Y}$ à chaque entité de $\mathcal{X}$ tiré selon la probabilité $\mathcal{P}$.

De nombreux algorithmes supervisés, dits **génératifs**, cherchent à apprendre un processus génératif permettant d'expliquer l'ensemble de données L, parmi lesquels on retrouve les approches de classification *naïve Bayesienne* ([Cestnik et al., 1987]), les *modèles de mixtures* ([Dempster et al., 1977]), les *machines de Boltzmann restreintes* ([Fischer and Igel, 2012]) ou *l'analyse linéaire discriminante* ([Fisher, 1938, Fukunaga, 1990]). Les algorithmes supervisés non génératifs sont dits **discriminants**. Parmi ces algorithmes, certains cherchent à apprendre un séparateur linéaire, tels que les *machines à vecteurs de support* ([Burges, 1998]) et les algorithmes basés sur des *perceptrons* ([Rosenblatt, 1962]). Les autres algorithmes supervisés, tels que les versions kernalisées des séparateurs linéaires, les *réseaux de neurones* ([Zhang, 2000]), ou les algorithmes basés sur la *logique* ([Murthy, 1998], [Fürnkranz, 1999]) cherchent eux à apprendre un séparateur non-linéaire pour la classification de jeu de données plus complexes. Des états de l'art plus complets l'apprentissage supervisé peuvent être trouvés, notamment [Kotsiantis, 2007].

Les algorithmes de classification supervisés basent leur apprentissage sur les données étiquetées disponibles. Or notamment à cause des contraintes de coûts et de difficulté d'étiquetage, les données non-étiquetées sont plus facilement accessible que les données étiquetées et la proportion d'entités étiquetées est souvent faible par rapport à l'ensemble des données disponibles. Les algorithmes supervisés ignorent donc un nombre non-négligeable de données dans le processus d'apprentissage et n'exploitent pas les informations portées par ces entités non-étiquetées. Les algorithmes transductifs exploitent les données étiquetées et non-étiquetées afin de prédire les étiquettes $\mathbf{y_U}$ des entités de U. La tâche n'est cependant résolue que pour les entités non-étiquetées disponibles pendant l'entraînement et le modèle ne peut pas être généralisé à de nouvelles entités. Les algorithmes semi-supervisés, inductifs, de classification exploitent quant à eux l'ensemble des données disponibles pendant l'entraînement, étiquetées et non-étiquetées, afin d'apprendre un modèle pour prédire l'étiquette de n'importe quelle entité dans $\mathcal{X}$ tirée selon la probabilité $\mathcal{P}$. Pour efficacement exploiter les données non-étiquetées disponibles, les algorithmes semi-supervisés supposent que la distribution sous-jacente aux données est structurée. Dans des contextes réels, deux entités proches sont plus susceptibles de partager des caractéristiques com-

munes que deux entités lointaines. L'hypothèse selon laquelle deux entités proches sont probablement semblables, **l'hypotèse d'homophilie**, est notamment introduite dans [McPherson et al., 2001]. Dans le cadre de l'apprentissage automatique, les hypothèses de **smoothness** et de **cluster** sont deux hypothèses sur la répartition des données basées sur l'hypothèse d'homophilie. Pour résoudre la tâche de classification, les algorithmes semi-supervisés supposent que les données manipulées satisfont l'une de ces hypothèses (comme décrit dans [Chapelle et al., 2006]).

De nombreuses approches semi-supervisées ont été proposées afin de résoudre la tâche de classification. Certains algorithmes utilisent les données non-étiquetées pour biaiser l'entraînement d'un algorithme de classification supervisé, parmi lesquels les algorithmes de **co-training** ([Blum and Mitchell, 1998]), de **self-training** ([Yarowsky, 1995]) et de **multi-view** ([de Sa, 1994]). Une partie des autres algorithmes semi-supervisés de classification peuvent être structurés similairement aux algorithmes supervisés. On retrouve donc des algorithmes semi-supervisés **génératifs**, tels que l'algorithme **espérance-maximisation** ([Dempster et al., 1977]), exploitant l'ensemble du jeu de données, sans considération pour l'étiquetage, pour estimer la densité sous-jacente des données, avant de prédire les étiquettes inconnues en fonction des étiquettes connues. De la même manière, les algorithmes semi-supervisés **disriminants** se basent sur l'ensemble des données pour apprendre un séparateur entre les classes, tels que dans les machines à vecteur de support transductifs ([Joachims, 1999]). Un dernier groupe d'algorithmes semi-supervisés résolvent la tâche en se basant sur les relations entre les entités plus que sur leur représentations vectorielles. Un graphe, composé par un ensemble d'entités et de lien, peut être utilisé pour représenter les entités d'un jeu de données et leurs relations. Une telle représentation d'un jeu de données peut facilement être construite et de nombreux algorithmes semi-supervisés, dit **basés sur les graphes**, exploitent la représentation d'un jeu de données sous forme de graphe pour résoudre la tâche. Une revue plus précise des graphes et d'algorithmes semi-supervisés basés sur les graphes est faite dans ces travaux. Des travaux plus complets sur l'apprentissage semi-supervisé sont disponibles dans différentes études, notamment [Zhu, 2005, Chapelle et al., 2006].

Les algorithmes d'*apprentissage automatique* sont appliqués sur des jeux de données afin d'apprendre un modèle ou d'extraire l'information voulue. Les données doivent donc d'abord être collectée à partir du contexte réel dans lequel le problème est étudié. Concernant les exemples tirés du site de commerce en ligne, les jeux de données peuvent par exemple être composés des clients, des informations collectées lors de leur inscription ou de leurs achats. La majorité des algorithmes d'*apprentissage automatique* se basent sur une représentation vectorielle des données. L'application de ces algorithmes nécessite donc la modélisation des données réelles au travers d'un vecteur de

description. A cause de la sélection des attributs par un expert et des transformations qui peuvent leur être appliquées, de nombreuses représentations vectorielles des données peuvent construites à partir des informations réelles disponibles. Que l'algorithme utilisé pour résoudre la tâche se base sur la représentation des données et les relations entre les entités, le résultat d'un algorithme dépend de la représentation vectorielle des données disponible sur laquelle se base directement ou indirectement l'algorithme et peut donc varier pour les différentes représentations vectorielles disponibles pour un même jeu de données. La modélisation vectorielle d'un jeu de données est donc une étape importante pour la résolution de la tâche ciblée.

La majorité des algorithmes d'*apprentissage automatique* calcule une distance ou une similarité entre les représentations vectorielles des données. Plusieurs distances et similarités classiques ont été définies et sont communément utilisées telles que la distance euclidienne ou la similarité cosinus. Pour le reste de ces travaux, nous considérerons des distances. Il faut cependant noter que les propos concernant les distance peuvent facilement être transposés pour les similarités en inversant l'échelle: de faibles (respectivement larges) distances peuvent être remplacées par de larges (respectivement faibles) similarités. Les distances classiquement utilisées considèrent que les différents attributs ont la même influence sur la tâche ciblée et les différents attributs participent donc de manière égale dans la proximité entre deux entités. A cause des choix arbitraires effectués lors des étapes menant à la représentation vectorielle des données, tous les attributs de la représentation vectorielle d'une entité n'ont pas la même influence pour la tâche à résoudre. L'hypothèse d'égale influence des attributs sur laquelle se basent les distances n'étant pas nécessairement satisfaite, les distances classiquement utilisées peuvent être sous-optimale pour la tâche considérée.

L'exécution d'un algorithme est dépendant de l'application de la distance sur la représentation vectorielle des données. La capacité de résolution d'un algorithme dépend donc du choix de la représentation vectorielle et de la distance. Dans la littérature, un ensemble de travaux cherchent à mutuellement adapter la représentation des données et la distance utilisée. Selon que l'algorithme se base sur la représentation vectorielle des données ou sur les relations, i.e. les distances, entre les entités, la représentation des données ou la distance entre les entités peuvent être appris. De nombreuses approches ont été proposées pour adapter la distance à la représentation vectorielle fixée des données (*apprentissage de métrique*) ou pour adapter la représentation vectorielle des données à la distance sélectionnée (*apprentissage de représentation*,*sélection d'attributs*,*réduction de dimension*).

Afin d'apprendre la distance reflétant au mieux la similarité entre les entités dans la représentation vectorielle disponible, la majorité des algorithmes d'apprentissage de métrique cherchent à apprendre les relations entre paires d'entités à partir d'un ensemble de contraintes. Bien que la majorité des

algorithmes d'apprentissage de métrique cherchent à apprendre une distance linéaire entre les entités, telle qu'une distance Mahalanobis dans [Weinberger and Saul, 2009], un certain nombres d'approches proposent d'apprendre une distance non-linéaire. Une distance non-linéaire peut être indirectement apprise en appliquant un algorithme d'apprentissage de métrique linéaire dans un espace de représentation non-linéaire ([Schölkopf et al., 1998, Scholkopf and Smola, 2001]). D'autres travaux, tels [Kedem et al., 2012] ou [Chopra et al., 2005], proposent des approches pour apprendre une distance non-linéaire en apprenant une nouvelle représentation des données. La non-linéarité peut être introduite dans la distance fixée ou dans la fonction de mapping apprise.

Ces métriques peuvent être apprises soit sur le domaine entier dans lequel les données sont tirées ou sur des patches locaux ([Frome et al., 2007, Ramanan and Baker, 2011]), afin de s'adapter à des données hétérogènes. Différentes études plus complètes sur l'apprentissage de métrique sont disponibles, notamment [Yang, 2006, Kulis, 2012, Bellet et al., 2013].

Bien que considérées comme des méthodes d'apprentissage de métrique, les algorithmes non-linéaires introduits précédemment peuvent être vus comme des algorithmes où les données sont projetées dans un nouvel espace dans lequel une distance fixée satisfait un ensemble de contraintes. Ces algorithmes peuvent donc être associés à l'*apprentissage de représentation*, qui cherche à projeter les données dans un espace de représentation dans lequel une distance est représentative de la tâche à résoudre. Selon le nouvel ensemble d'attributs qu'ils proposent, les algorithmes d'apprentissages peuvent être regroupés en différentes catégories.

Les premiers, appelés algorithmes de *sélection d'attributs* ([Guyon and Elisseeff, 2003]), choisissent un sous-ensemble des attributs initiaux, selon leurs capacités de prédiction individuelles ou combinées, tels que les algorithmes basés sur les arbres ([Geurts et al., 2006]). Les attributs initialement disponibles pouvant être peu adaptés à la tâche ciblée, les autres algorithmes d'apprentissage de représentation proposent de construire un ensemble d'attributs en transformant les attributs initialement disponibles, tout en réduisant le nombre d'attributs décrivant les données. Ces approches sont dans le domaine de la *réduction de dimensionnalité*, parmi lesquels on retrouve l'*Analyse en Composantes Principales*, ou les auto-encoders ([Rifai et al., 2011]). Un dernier groupe d'algorithmes d'apprentissage de représentation proposent un nouvel ensemble d'attributs sans contraintes sur la dimension. Avec ces méthodes, les données sont projetées dans un nouvel espace de représentation en appliquant des transformations sur l'ensemble initial d'attributs. Parmi ces algorithmes et proches de l'approche proposée dans ces travaux, les algorithmes développés dans [J. Weston, 2008] et [Hoffer and Ailon, 2014] apprennent un réseau de neurones pour projeter les données dasn un espace de représentation satisfaisant certaines contraintes liées à une distance.

Différentes études traitant de manière plus précise *l'apprentissage de re-*

présentation sont disponibles, notamment [Bengio et al., 2013, Guyon and Elisseeff, 2003].

Une fois que la représentation vectorielle du jeu de données et la distance ont été fixées, un graphe peut être construit à partir du jeu de données pour résoudre la tâche ciblée. Une partie des algorithmes semi-supervisés se basent sur des données structurées en graphes pour résoudre les différentes tâches.

Un graphe $G = (V, E)$ est composé d'un ensemble de nœuds V qui sont liés par un ensemble de liens E qui peuvent être pondérés. Les poids utilisés pour caractériser les liens permettent de représenter une relation entre les nœuds liés par les liens. Un graphe $G = (X, E)$ peut être construit à partir d'un jeu de données X. Les liens de E entre les paires d'entités de X peuvent par exemple être créés en calculant la distance entre les représentations vectorielles des entités.

La représentation sous forme d'un graphe d'un jeu de données permet de faire ressortir une structure au sein des données, par l'ensemble des relations entre les entités qui sont représentées. Certains liens présents dans le graphe sont associés à des poids faibles ou ne portent que peu d'information par rapport aux autres liens. Or nous cherchons à renforcer la structure au sein des données que le graphe a fait ressortir. L'objectif est donc de ne conserver que les liens les plus représentatifs des relations entre les instances ou les liens les plus forts. Les liens négligeables d'un graphe doivent donc être supprimés. Un lien peut être considéré négligeable si le poids qui lui est associé est négligeable par rapport à l'ensemble du graphe. Un seuil est donc appliqué sur l'ensemble des liens pour définir si les liens sont conservés ou non, i.e. la **simplification** ϵ. Un lien peut au contraire être considéré comme négligeable comparé à l'ensemble des liens associés aux entité communes. Pour une entité donnée du graphe, seuls les k liens les plus forts sont conservés, i.e. la **simplification** k**-nn**. A partir d'un même jeu de données X, de nombreux graphes peuvent donc être construits selon le choix de la distance pour pondérer les liens et de la méthode pour simplifier le graphe.

Différents algorithmes semi-supervisés ont été développés pour résoudre une tâche de classification en exploitant des données représentées sous forme d'un graphe. Parmi ces algorithmes, les auteurs de [Zhu and Ghahramani, 2002] proposent un algorithme itératif pour résoudre la tâche de classification en se basant sur le graphe représentant le jeu de données. A chaque étape de l'algorithme, chaque nœud propage son étiquette actuelle à l'ensemble de ces voisins. L'étiquette d'une entité initialement non-étiquetée est donc la moyenne des étiquettes de ses voisins, pondérées par leur proximité. Plusieurs variations de cet algorithme et approches proches peuvent également être trouvées ([Zhu et al., 2003, Zhou et al., 2004, Belkin et al., 2004]).

Ces algorithmes sont cependant dépendant du graphe disponible et plus précisément sur les choix effectué lors de la construction du graphe, notamment le choix de la distance pour pondérer les liens. Différents travaux

ont été menés pour étudier l'influence de la méthode de construction du graphe sur des algorithmes d'apprentissage automatique, notamment dans [Maier, Markus et al., 2013] et [de Sousa et al., 2013].

L'exécution, et donc le résultat, des algorithmes de classification dépendent de l'adéquation entre la représentation vectorielle des données et la distance utilisée. La question principale sous-jacente à ces travaux est donc liée à la méthode à utiliser pour adapter au mieux la représentation vectorielle des données et la distance associée. Comment la représentation vectorielle et la distance associée peutvent être apprise de sorte à ce que la distance soit représentative de la tâche à résoudre? Les approches d'apprentissage de métrique et de représentation sont liées. Dans ces travaux, en considérant une distance prédéfinie, nous étudions les méthodes d'apprentissage de représentation pour la résolution de tâche en se basant sur des graphes, en appliquant l'algorithme de propagation d'étiquettes introduit dans [Zhu and Ghahramani, 2002]. Nous affirmons qu'un algorithme d'apprentissage de représentation guidé par la tâche permet aux algorithmes semi-supervisés basés sur les graphes de résoudre la tâche de manière plus optimale qu'avec la représentation initiale des données.

Dans ces travaux, nous cherchons à construire un graphe sur lequel l'algorithme de propagation d'étiquettes peut être appliqué de manière optimal. Cet algorithme de classification est optimal si les entités proches dans le graphe sont similaires au regard de la tâche à résoudre. Afin de construire un graphe à partir d'un jeu de données X tel que les entités proches ont une étiquette similaire, une distance d_ϕ doit être apprise telle que deux entités avec la même étiquette sont plus proches l'une de l'autre que d'une entité étiquetée différement. Plus formellement, la distance d_ϕ doit être apprise telle que $d_\phi(x_i, x_j) < d_\phi(x_i, x_k)$ pour le plus de triplets d'entités $(x_i, x_j, x_k) \in X \times X \times X$ tels que $y_i = y_j \neq y_k$ possible. Pour apprendre une telle distance, nous proposons d'apprendre une nouvelle représentation $\phi(x)$ des entités $x \in X$ telle que pour une distance d fixée, $d(\phi(x_i), \phi(x_j)) < d(\phi(x_i), \phi(x_k))$ pour le plus grand nombre de triplets (x_i, x_j, x_k) tels que $y_i = y_j \neq y_k$ possible. Définissons T comme étant l'ensemble des triplets:

$$T = \{(x_i, x_j, x_k) | x_i, x_j, x_k \in X \wedge y_i = y_j \neq y_k\}$$

La nouvelle représentation ϕ des données peut être apprise en minimisant la fonction suivante

$$C(\phi|T) = \sum_{(x_i,x_j,x_k)\in T} max(0, \mu - [d(\phi(x_i), \phi(x_j) - d(\phi(x_i), \phi(x_k))]) \quad (1)$$

Les réseaux de neurones artificiels, de par leur fortes capacités de représentation ([Hornik et al., 1989]), sont de plus en plus fréquemment utilisés pour des problèmes d'apprentissage de représentation.

Un réseau de neurones est composé d'un ensemble de couches, elles même composée d'un ensemble d'unités appelées neurones. Un neurone est une unité produisant une valeur continue obtenue en appliquant une fonction non-linéaire, dite d'activation, sur une combinaison pondérée de ses entrées. C'est une généralisation du concept de perceptron ([Rosenblatt, 1958]) qui produit une sortie binaire par application d'un seuil sur une combinaison pondérée de ses entrées. Les entrées de n'importe quel neurone d'un perceptron multi-couches est l'ensemble, ou un sous-ensemble, des sorties produites par les unités de la couche précédente. Les perceptrons multi-couches sont donc un sous-ensemble des réseaux de neurones tels que chaque couche d'un perceptron multi-couche n'est connecté qu'à la couche précédente et suivante, sans cycle dans le réseau. Un perceptron multi-couches associe donc une nouvelle description $\phi(x)$ à une entité donnée x. Les réseaux de neurones peuvent être facilement optimiser grâce à la méthode de **rétro-propagation de l'erreur** ([Lecun, 1985, Rumelhart et al., 1986, Le Cun, 1986]) qui modifie chacun des poids du réseau en fonction de son influence relative sur la sortie produite. La représentation ϕ apprise par notre algorithme sera donc un perceptron multi-couches. De tels réseaux de neurones ne considèrent cependant que des entités seules, quand nous nous intéressons à des relations entre entités. Afin de travailler sur des relations entre les entités, une architecture siamoise est construite à partir de ϕ pour optimiser notre fonction de coûts. Initialement proposé dans [Bromley et al., 1994], les architectures siamoises permettent d'appliquer simultanément le même réseau de neurones sur différentes entités distinctes données en entrée. Un réseau de neurones siamois est une architecture composée d'autant de sous-réseaux clones qu'il y a d'entrée distinctes. Un comparateur est ajouté au dessus des réseaux dupliqués pour comparer la sortie obtenue de chacun d'entre eux, afin d'évaluer la fonction de coût finale.

Nous appellerons l'algorithme d'apprentissage de représentation introduit ici *MDRL* pour *Metric Driven Representation Learning*. Une fois que la nouvelle représentation des données est apprise en optimisant la fonction introduite dans l'équation 1, un graphe peut être construit afin de résoudre la tâche de classification en appliquant l'algorithme de propagation d'étiquettes. Le processus global pour la résolution de la tâche de classification est résumé

dans l'algorithme 1.

Algorithm 1: MDRL: Processus global pour la classification basé sur l'apprentissage de représentation

Data: (L, y_L) un ensemble d'exemples étiquetés et de leurs étiquettes associées
U un ensemble d'entités non-étiquetées
d une distance
Result: $\mathbf{y_U}$ un vecteur de prédictions pour les entités dans U

1 $\phi \leftarrow \text{MDRL}(L, y_L, d)$ `// Apprentissage de la représentation`
2 $\hat{X} \leftarrow \phi(X)$ `// Projection du jeu de données dans le nouvel espace de représentation`
3 $G, W \leftarrow \text{Graph Construction}(\hat{X}, d)$ `// Construction du graphe dans le nouvel espace de représentation`
4 $y_U \leftarrow \text{Label Propagation}(W, y_L)$ `// Classification des entités non-étiquetées`

Dans la suite de cette thèse, nous prouvant notre affirmation en montrant qu'un graphe tel que seules les entités similairement étiquetées sont liées peut être construit. L'application de l'algorithme de propagation d'étiquette sur un tel graphe est assuré de résoudre la tâche de manière optimale. Dans l'analyse théorique, nous prouvons qu'une valeur spécifique d'ϵ peut être définie afin d'appliquer une simplification ϵ sur le graphe construit dans la nouvelle représentation des données grâce à l'algorithme **MDRL**. L'existence de cet ϵ est assurée si les entités non-étiquetées se trouvent dans un voisinage suffisamment proche d'au moins une entité similairement étiquetées. La taille du voisinage peut être défini en fonction de la représentation ϕ apprise par l'algorithme **MDRL** et les exemples étiquetées du jeu de données.

Pour prouver l'existence d'un tel ϵ, un premier lemme montre que la distance entre deux entités dans l'espace de représentation associé à ϕ est bornée en fonction de leur distance dans l'espace initial et de la fonction ϕ. Un deuxième lemme permet de prouver que si les entités d'un premier triplet sont dans le voisinage des entités étiquetées similairement composant un deuxième triplet satisfaisant la contrainte relative associée, alors le premier triplet satisfait la contrainte relative qui lui est associée. En combinant ces premiers lemmes, un troisième lemme définit les conditions nécessaires pour qu'un triplet d'entités non-étiquetées puisse satisfaire la contrainte relative associée dans le nouvel espace de représentation associée à ϕ. En se basant sur ce troisième lemme, une proposition définit la valeur d'ϵ permettant d'obtenir un graphe tel que seules les entités identiquement étiquetées sont liées. Notre théorème principal permet donc d'assurer une classification optimale.

L'objectif de l'algorithme d'apprentissage de représentation introduit est de projeter les données dans un espace vectoriel dans lequel la tâche de clas-

sification peut être résolue de manière plus optimale que dans l'espace de représentation initial. Afin d'évaluer la pertinence de l'algorithme proposé dans ces travaux, différentes expériences ont été menés sur divers jeux de données artificiels et réels. Les résultats obtenus sont comparés à ceux obtenus pour d'autres algorithmes d'apprentissage de métrique ou de représentations.

Dans une première étape, la satisfaction des contraintes associées à la distance suite à l'application de l'algorithme d'apprentissage de représentation *MDRL* est évaluée sur différents jeux de données en calculant le pourcentage de triplets pour lesquels la contrainte associée est satisfaite. A l'issue de cette expérience, on a pu observer que la représentation apprise par l'algorithme *MDRL* permet de minimiser le nombre de triplets pour lesquelles la contrainte associée est non-satisfaite par rapport aux autres algorithmes. Dans un second temps, la satisfaction des hypothèses sur lesquelles se base l'analyse théorique est empiriquement évaluée. On a ainsi pu observer empiriquement que les hypothèses ne sont pas nécessairement satisfaites pour les différents jeux de données. Nous évaluons empiriquement ensuite le gain de l'algorithme *MDRL* pour la résolution de la tâche de classification. Pour se faire l'erreur de classification est comparée pour les différents paires de représentation vectorielle et de distance. Il ressort de ces expériences que l'erreur de classification obtenue dans l'espace appris par l'algorithme *MDRL* est parmi les plus faibles pour les différents jeux de données. L'analyse théorique se concentre sur une seule méthode de construction de graphe et un seul algorithme de classification. Différentes expériences ont donc été réalisées afin d'évaluer la pertinence de l'algorithme *MDRL* pour ces différents cadres de résolution de la classification. Dans ce contexte également, l'erreur de classification obtenue dans l'espace appris par l'algorithme *MDRL* est parmi les plus faibles pour les différents jeux de données.

Dans ces travaux, nous nous sommes intéressé à la résolution d'une tâche de classification par une approche basée sur des graphes. Afin d'appliquer l'algorithme de propagation d'étiquettes, un graphe doit être construit à partir de la représentation vectorielle des données. Nous avons proposé l'algorithme *MDRL* afin de projeter les données dans un espace permettant la construction d'un graphe le plus adapté possible à la propagation d'étiquettes. Une première analyse théorique a permis de définir des conditions initiales sur les données telles qu'une classification optimale peut être assurée. Les différentes évaluations empiriques qui ont été faites ont montré l'intérêt d'une telle approche pour améliorer les résultats de la classification, malgré le fait que les hypothèses théoriques ne soient pas nécessairement satisfaites. La non-satisfaction de hypothèses théoriques et l'intérêt de l'algorithme *MDRL* pour la résolution de la tâche de classification pour d'autres méthodes de construction de graphes et de classification nous permet d'envisager une relaxation ou une extension de l'analyse théorique menée dans ces travaux. Dans cette optique, différents axes de travail sont proposés,

afin de proposer des garanties plus générales.

Bibliography

[Belkin et al., 2004] Belkin, M., Matveeva, I., and Niyogi, P. (2004). Regularization and semi-supervised learning on large graphs. In *In COLT*, pages 624–638. Springer.

[Bellet et al., 2013] Bellet, A., Habrard, A., and Sebban, M. (2013). A Survey on Metric Learning for Feature Vectors and Structured Data. *ArXiv e-prints*.

[Bengio et al., 2013] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA. ACM.

[Bromley et al., 1994] Bromley, J., Guyon, I., Lecun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a "siamese" time delay neural network. In *In NIPS Proc.*

[Burges, 1998] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

[Cestnik et al., 1987] Cestnik, G., Konenenko, I., and Bratko, I. (1987). Assistant-86: A knowledge-elicitation tool for sophisticated users. In Bratko, I. and Lavrac, N., editors, *Progress in Machine Learning*, pages 31–45. Sigma Press, Ljubljana.

[Chapelle et al., 2006] Chapelle, O., Schlkopf, B., and Zien, A. (2006). *Semi-Supervised Learning*. The MIT Press, 1st edition.

[Chopra et al., 2005] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 -*

Volume 01, CVPR '05, pages 539–546, Washington, DC, USA. IEEE Computer Society.

[de Sa, 1994] de Sa, V. R. (1994). Learning classification with unlabeled data. *Advances in neural information processing systems*, pages 112–112.

[de Sousa et al., 2013] de Sousa, C. A. R., Rezende, S. O., and Batista, G. E. A. P. A. (2013). Influence of graph construction on semi-supervised learning. In Blockeel, H., Kersting, K., Nijssen, S., and Zelezný, F., editors, *ECML/PKDD (3)*, volume 8190 of *Lecture Notes in Computer Science*, pages 160–175. Springer.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.

[Fischer and Igel, 2012] Fischer, A. and Igel, C. (2012). *An Introduction to Restricted Boltzmann Machines*, pages 14–36. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Fisher, 1938] Fisher, R. A. (1938). The statistical utilization of multiple measurements. *Annals of Eugenics*, 8(4):376–386.

[Frome et al., 2007] Frome, A., Singer, Y., Sha, F., and Malik, J. (2007). Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*, pages 1–8.

[Fukunaga, 1990] Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition (2Nd Ed.)*. Academic Press Professional, Inc., San Diego, CA, USA.

[Fürnkranz, 1999] Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54.

[Geurts et al., 2006] Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Mach. Learn.*, 63(1):3–42.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

[Hoffer and Ailon, 2014] Hoffer, E. and Ailon, N. (2014). Deep metric learning using triplet network. *CoRR*, abs/1412.6622.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366.

[J. Weston, 2008] J. Weston, F. Rattle, R. C. (2008). Deep learning via semi-supervised embedding. In *International Conference on Machine Learning*.

[Joachims, 1999] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 200–209, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Kedem et al., 2012] Kedem, D., Tyree, S., Weinberger, K., Sha, F., and Lanckriet, G. (2012). Non-linear metric learning. In Bartlett, P., Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 2582–2590.

[Kotsiantis, 2007] Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. *Informatica (Slovenia)*, 31(3):249–268.

[Kulis, 2012] Kulis, B. (2012). Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364.

[Le Cun, 1986] Le Cun, Y. (1986). *Learning Process in an Asymmetric Threshold Network*, pages 233–240. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Lecun, 1985] Lecun, Y. (1985). *Une procedure d'apprentissage pour reseau a seuil asymmetrique (A learning scheme for asymmetric threshold networks)*, pages 599–604.

[Maier, Markus et al., 2013] Maier, Markus, von Luxburg, Ulrike, and Hein, Matthias (2013). How the result of graph clustering methods depends on the construction of the graph. *ESAIM: PS*, 17:370–418.

[McPherson et al., 2001] McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444.

[Murthy, 1998] Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389.

[Ramanan and Baker, 2011] Ramanan, D. and Baker, S. (2011). Local distance functions: A taxonomy, new algorithms, and an evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):794–806.

[Rifai et al., 2011] Rifai, S., Dauphin, Y., Vincent, P., Bengio, Y., and Muller, X. (2011). The manifold tangent classifier. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 2294–2302.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386.

[Rosenblatt, 1962] Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington. it Early work on what would now be referred to as a "connectionist" model.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA.

[Schölkopf et al., 1998] Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319.

[Scholkopf and Smola, 2001] Scholkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

[Weinberger and Saul, 2009] Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research (JMLR)*, 10:207–244.

[Yang, 2006] Yang, L. (2006). Distance metric learning: A comprehensive survey.

[Yarowsky, 1995] Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Zhang, 2000] Zhang, G. P. (2000). Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462.

[Zhou et al., 2004] Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In Thrun,

S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 321–328. MIT Press.

[Zhu, 2005] Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

[Zhu and Ghahramani, 2002] Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University.

[Zhu et al., 2003] Zhu, X., Ghahramani, Z., and Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 912–919.