





LUMIERE UNIVERSITY LYON II

Doctoral School InfoMaths (EDA 512)

A DISSERTATION

Submitted for the Degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

A Contribution to Topological Learning and its Application in Social Networks

by

Diala Ezzeddine

ERIC Laboratory, Lumiere University Lyon II

Under the Supervision

of

Djamel A. Zighed And **Fabien Rico**

1 October 2014

Committee

Mr. Gilles VENTURINI	Rapporteur	Professor, University of Tours
Mrs. Rosanna VERDE	Rapporteur	Professor, Second University of Naples
Mrs. Christine LARGERON	Examiner	Professor, Jean Monnet University St-Etienne
Mr. Djamel A. ZIGHED	Supervisor	Professor, Lumiere University Lyon II
Mr. Fabien RICO	Co-Supervisor	A. Professor, University Claude Bernard Lyon I

Abstract

Supervised Learning is a popular field of Machine Learning that has made recent progress. In particular, many methods and procedures have been developed to solve the classification problem. Most classical methods in Supervised Learning use the density estimation of data to construct their classifiers.

In this dissertation, we show that the topology of data can be a good alternative in constructing classifiers. We propose using topological graphs like Gabriel graphs (GG) and Relative Neighborhood Graphs (RNG) that can build the topology of data based on its neighborhood structure. To apply this concept, we create a new method called **Random Neighborhood Classification (RNC)**. In this method, we use topological graphs to construct classifiers and then apply Ensemble Methods (EM) to get all relevant information from the data. EM is well known in Machine Learning, generates many classifiers from data and then aggregates these classifiers into one. Aggregate classifiers have been shown to be very efficient in many studies, because it leverages relevant and effective information from each generated classifier.

We first compare RNC to other known classification methods using data from the *UCI* Irvine repository. We find that RNC works very well compared to very efficient methods such as Random Forests and Support Vector Machines. Most of the time, it ranks in the top three methods in efficiency. This result has encouraged us to study the efficiency of RNC on real data like tweets.

Twitter, a microblogging Social Network, is especially useful to mine opinion on current affairs and topics that span the range of human interest, including politics. Mining political opinion from Twitter poses peculiar challenges such as the versatility of the authors when they express their political view, that motivate this study. We define a new attribute, called couple, that will be very helpful in the process to study the tweets opinion. A couple is an author that talk about a politician. We propose a new procedure that focuses on identifying the opinion on tweet using couples. We think that focusing on the couples's opinion expressed by several tweets can overcome the problems of analysing each single tweet. This approach can be useful to avoid the versatility, language ambiguity and many other artifacts that are easy to understand for a human being but not automatically for a machine.

We use classical Machine Learning techniques like KNN, Random Forests (RF) and also our method RNC. We proceed in two steps: First, we build a reference set of classified couples using Naive Bayes. We also apply a second alternative method to Naive method, sampling plan procedure, to compare and evaluate the results of Naive method. Second,

we evaluate the performance of this approach using proximity measures in order to use RNC, RF and KNN. The expirements used are based on real data of tweets from the French presidential election in 2012. The results show that this approach works well and that RNC performs very good in order to classify opinion in tweets.

Topological Learning seems to be very intersting field to study, in particular to address the classification problem. Many concepts to get informations from topological graphs need to analyse like the ones described by Aupetit, M. in his work (2005). Our work show that Topological Learning can be an effective way to perform classification problem.

Keywords: Machine Learning, Classification, Neighborhood Graphs, Gabriel Graph, Relatif Neighborhood Graph, Ensemble Methods, Naive Bayes, Decision Templates, Twitter, Opinion Mining.

Acknowledgements

I would like to acknowledge and give my sincerest thanks to the many people without whom this work would not have been possible.

My supervisor Djamel Zighed gave valuable guidance and advice. I really appreciated your wisdom, enthusiasm, and energy throughout.

I am grateful for having had Fabien Rico as my Second Supervisor. He introduced me to the world of programming. His counsel, understanding, and kind patience were invaluable every step of the way. For everything you did for me, Fabien, I sincerely thank you.

Professors Gilles Venturini, Rosanna Verde, and Christine Largeron expressed interest in this research and honored me with their service on the committee.

Members of the ERIC laboratory made me feel at home with their help and support. I want to especially thank Prof. Jean-Hugues Chauchat for his attention and encouragement. I am so grateful to have had the chance to know you all. Thank you Julien, Olena, Jihene, Mathilde, Marouane, Vincent, Marian-Andrei, Huu-hoa and Rashed for your friendship that eased the years of being away from my home and family.

Mr. Antoine Rolland and Jihene gave good advice and collaborated with me on the Twitter work.

Professors Rafik Abdesselam, Helene Lample, Anne Perrut and Joel Gaden gave me the precious opportunity to teach along with vital help and advice. I really enjoyed working with you all.

The Lebanese University gave me the scholarship that realized my dreams. Thank you Prof. Ali Mneimneh for believing in me. My deepest gratitude goes to Prof. Addullah Hakim for helping me come to France and ensuring the continued financial support of my thesis through the university.

My good friend Rada gave so much of her time, support, and kindness. Your friendship is priceless to me. Thank you from all my heart.

My family: my father Mohamad; my mother Wafaa; my sisters, Dima, Dunia, and Diana; and my brothers Wassim and Wissam. Thank you for your unlimited love, support, and faith in me.

Finally, I would like to thank my husband, Ibrahim. Your love, support, patience, and encouragement were precious to me and irreplaceable in completing this thesis. Thank you for believing in me and for helping me follow my dreams.

To Mohamad and Ibrahim...

Contents

Abstract iii

1 Introduction 1

2 Machine Learning 3
2.1 Introduction . . . 3
2.1.1 Notations . . . 4
2.1.2 Classifier . . . 5
2.2 Ensemble Methods . . . 7
2.2.1 Bagging . . . 8
2.2.2 Boosting . . . 9
2.2.3 AdaBoost . . . 10
2.3 K Nearest Neighbors . . . 11
2.4 Learning Trees . . . 13
2.4.1 Decision tree . . . 13
2.4.2 C4.5 . . . 15
2.4.3 Random Forests . . . 16
2.5 Support Vector Machine . . . 18
2.5.1 Kernel . . . 19
2.5.2 Linear SVM . . . 20
2.5.3 Non linear SVM . . . 24
2.5.4 Soft Margin Hyperplane . . . 27
2.5.5 Multi-class SVM . . . 29
2.5.5.1 One-Against-All . . . 29
2.5.5.2 One-Against-One . . . 30
2.6 Linear Probabilistic Model . . . 30
2.6.1 Logistic Regression . . . 31
2.6.2 Discriminant Analysis . . . 32
2.7 Quality Measures . . . 33
2.7.1 Kernel Target Alignment (KTA) . . . 34
2.7.2 Feature Space-based kernel Matrix evaluation measure (FSM) . . . 35
2.7.3 Relative Cut Edge Weight (RCEW) . . . 35
2.8 Statistical Tests . . . 37
2.8.1 Wilcoxon Signed Ranks test . . . 37

2.8.2 5x2cv Paired test . . . 38
2.9 Conclusion . . . 39

3 Neighborhood Random Classification 41
3.1 The General Framework . . . 43
3.1.1 Random Neighborhood Classification . . . 43
3.1.1.1 Sampling procedures . . . 44
3.1.1.2 Aggregting Functions . . . 44
3.1.2 Example: Random Forests . . . 45
3.2 Neighborhood Graphs . . . 45
3.2.1 Neighborhood structure . . . 46
3.2.2 Local individual classifiers . . . 47
3.2.3 Neighborhood Graph . . . 48
3.2.3.1 The Minimum Spaning Tree (MST) . . . 49
3.2.3.2 The Relative Neighborhood Graph (RNG) . . . 50
3.2.3.3 The Gabriel Graph (GG) . . . 51
3.2.3.4 The Delaunay Triangulation (DT) . . . 52
3.2.4 Partition by neighborhood graphs . . . 52
3.3 Aggregation Methods . . . 54
3.3.1 Simple methods . . . 57
3.3.2 Weighted majority or mean . . . 58
3.3.3 Naive Bayes (NB) . . . 59
3.3.4 Decision Templates (DecT) . . . 60
3.4 Evaluations . . . 61
3.4.1 Implementation of Random Topological Classifier (RTC) . . . 62
3.4.2 Implementation of the other methods . . . 63
3.4.3 Validation process . . . 63
3.4.4 Computational analysis . . . 64
3.4.5 The choice of the appropriate statistical test . . . 67
3.4.6 Why Delaunay is difficult to use ? . . . 70
3.5 Conclusion . . . 71

4 Detection of Political Opinion in Tweets 73
4.1 Introduction . . . 73
4.2 Related Work . . . 74
4.3 Problem Statement . . . 76
4.4 Approach Overview . . . 78
4.4.1 Creating the Reference Set . . . 78
4.4.1.1 Bayes method . . . 80
4.4.1.2 Sampling procedures . . . 82
4.4.2 Label Generalization . . . 84
4.5 Evaluation and Results . . . 85
4.5.1 Corpus . . . 85
4.5.2 Validation of the Naive Bayes labeling procedure and the Sampling methods . . . 86
4.5.3 Why these specific profiles? . . . 86
4.5.4 Results . . . 87

4.6 Conclusion 89

5 Conclusion **91**
5.1 Thesis Review and Conclusions 91
5.2 Ideas for future work 93

Appendix **95**

A Machine Learning **95**
A.1 Evaluation measure 95
A.2 Dataset Example 96
A.3 Suitable statistical test 96
A.4 Wilcoxon Test 99

B Detection of Political Opinion in Tweets **101**
B.1 The original French Tweets Examples 101

Bibliography **103**

Résumé **117**

List of Figures

1.1 Neighborhood graphs . . . 2

2.1 Scatter plot of Ω' in $\mathbb{R}^2$ with two classes . . . 6
2.2 The EM general algorithm . . . 8
2.3 Simple example that illustrate the Boosting Algorithm . . . 10
2.4 K nearest Neighbours (k=5) . . . 12
2.5 **playTennis** Example . . . 14
2.6 The Random Forests Procedure . . . 17
2.7 Non linear separator . . . 20
2.8 linear separator using the kernel k . . . 21
2.9 Representation of the Data set . . . 22
2.10 Optimal hyperplane . . . 23
2.11 Two seperators: linear and non linear hyperplane . . . 26
2.12 Overfitting caused by a non linear hyperplane: *$C = 1$ (up left), $C=10$ (up right), $C=100$ (down left) and $C=1000$ (down right)* . . . 27
2.13 Soft margin hyperplane . . . 28
2.14 A simple example, Multi-class SVM . . . 29
2.15 Data in the direction between-class centers . . . 36

3.1 The EMs based on neighborhood graph . . . 43
3.2 Partition of the representation space with a tree and the induced neighborhood structure. Each node gives the distribution of the most efficient predicted classes . . . 48
3.3 Minimum Spanning Tree Graph of Ω' . . . 50
3.4 Relative Neighbors Graph of Ω' . . . 50
3.5 Gabriel graph of Ω' . . . 51
3.6 Delaunay Triangulation Graph of Ω' . . . 52
3.7 The Delaunay Triangulation Graph of Ω' defined by the Voronoï structure 53
3.8 Four different Neighborhood Graphs: the neighbor structures of the Relative Neighbors graph, the GABRIEL graph, the Minimum Spanning Tree graph and the Delaunay graph, using the dataset Ω' introduced in table 2.1 54
3.9 Partition by neighborhood graphs . . . 55
3.10 The three decision rule to affect a new point ω_{new} using Gabriel Graph . 56
3.11 DP(ω) . . . 57
3.12 The simple aggregation methods . . . 58

4.1 The two parts of our approach . . . 79
4.2 Strategy used in the Progressive sampling procedure . . . 83
4.3 The goal of the Label Generalization is labeling a new tweet's author . . . 85

4.4 Words of all tweets written by authors **With** Sarkozy (**Against** Hollande) 88
4.5 Words of all tweets written by authors **With** Hollande (**Against** Sarkozy) 89

A.1 Choice of statistical test based on the data description 97

List of Tables

2.1 Set of points in R^2 with two classes . . . 5
2.2 Examples of kernel functions . . . 25
2.3 Setting up the wilcoxon signed ranks test . . . 38
2.4 Test error . . . 39

3.1 Confusion matrices of respectively ϕ_1, ϕ_2 and ϕ_3 . . . 59
3.2 Naive Bayes Example: probability calculations . . . 59
3.3 Results 1: Error rate of applying **RTC** classifiers based on the three neighborhood graphs RNG, GG and MST with respect to several aggregation functions . . . 65
3.4 Results 2: Error rate of applying **RTC** classifiers based on the three neighborhood graphs RNG, GG and MST with respect to several aggregation functions . . . 66
3.5 The Mean Error and the Mean Rank for all the different **RTC** based of neighborhood graphs compared in tables 3.3 and 3.4 . . . 67
3.6 Comparison of $\mathbf{RTC}_{RNG}$ to several classification algorithms . . . 68
3.7 Mean Error and Mean Rank of all the methods used in table 3.6 . . . 69
3.8 Mean of the number of neighbors and the RCEW index find out by MST, RNG and GG graphs . . . 71

4.1 Progressive sampling plan . . . 83
4.2 Description of the two databases used in this work . . . 85
4.3 Comparing the manual labels with Naive Bayes and sampling labels . . . 86
4.4 Three different profiles . . . 87
4.5 The corresponding Accuracy of the three confusion matrix calculating using the three different profiles cases . . . 87
4.6 Results of Leave one out procedure . . . 89

A.1 Contingency table that represents the classification results of an experiment 95
A.2 Dataset Example . . . 96
A.3 Statistical tests based on the type of the dependent and independent variables . . . 98

Chapter 1

Introduction

Machine Learning, the science of learning from data, has increasingly developed over the past two decades. Usually, Machine learning is divided into two main types: the **supervised learning** and the unsupervised learning. In this dissertation, we are concerned with the supervised learning part of Machine Learning, known also as predictive learning. More specifically, we are studying a particular form of the supervised learning, the **classification** problem.

Suppose we need to develop a program that, when given a picture of a person, can determine whether the person is male or female. In this case, the problematic of classification is to construct the best program based on a given set of classified pictures. This program is called a classier, because it assigns a class, male or female, to an object, a picture. Formally, this study consists to *learn a mapping from inputs* $\mathbf{x}$ *(pictures) to categorical outputs y (male, female) based on pre-labeled set of input-output pairs*, called training dataset. In this learning field, to construct this classifier, one of the following concepts is used:

- The density estimation of the training data;
- The topology of the training data;
- The morphology of the training data;

Most classical methods in supervised learning, whether probabilistic, statistical or geometric, essentially use the first concept. In this work, we want to examine the use of the topological concept in order to improve the result of the supervised learning approachs. To study this concept, we propose to use geometric graphs that build the topology of the training data. For instance, **Gabriel graphs** (Preparata, F. P. and Shamos, M.I (1985) [121]) or **Relative Neighborhood graphs** (RNG) (Toussaint, G.T(1980) [147]),

provide a neighborhood structure that express, in some way, the topological structure of the training data in multidimensional space. The figure 1.1 shows an example of topological structure based on Relative Neighborhood graphs and Gabriel graphs.

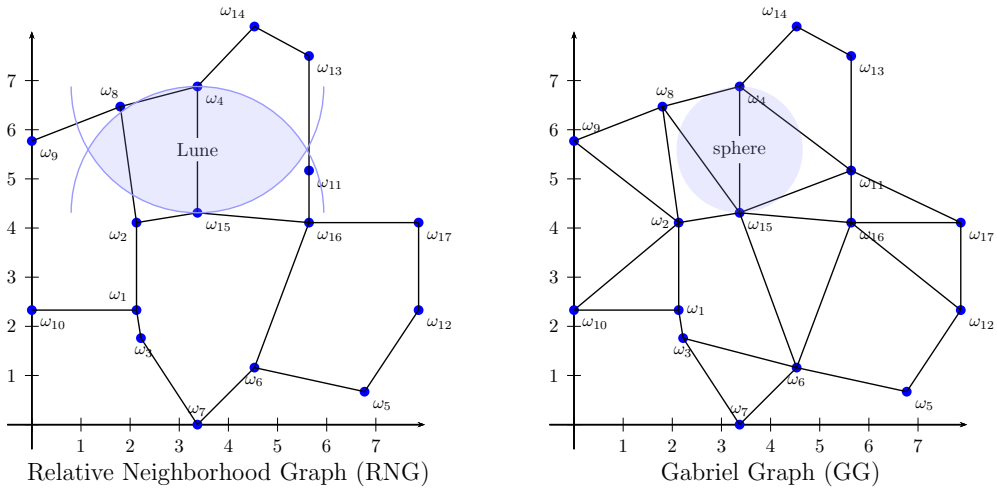

FIGURE 1.1: Neighborhood graphs

In the literature, there are many different learning algorithms that have been used to solve the supervised classification problem. **Ensemble methods (EM)** are one of the very known procedure in this field. They are more and more popular today due to their efficiency in prediction. EMs are based on multiple classifiers generated in random space from a training set. As consequence of the random sampling, it can be noted that the representation space improves along the process. This improvement leads to a better classification.

The structure of this Thesis

The work for this dissertation is three-fold. First, we provide an overview over the vast range of methods and procedures used in supervised learning. In the second part, we propose a new method, called **Random Neighborhood Classification (RNC)**, that performs very well in comparison to many other very known methods such as Random Forests and Support Vector Machine. This method combines between the Ensemble methods procedure and the topological graphs. In chapter 3, we explain and evaluate the RNC method implemented using the statistical **R** software. For the third part of this dissertation, we propose a new procedure to detect informations from tweets in order to analyse the political opinion in **Twitter**. Based on this procedure, we can classify tweets easly using any learning method. Chapter 4 provides the details of this new procedure and the results of using our **RNC** method.

Chapter 2

Machine Learning

2.1 Introduction

One of human characteristic is the ability to learn from everything, organize and classify anything in the world. These tasks become widely difficult to be analysed by a human especially when it contains a big number of examples. Therefore, Many learning algorithms are created to help human for more understanding and knwoledge of many things around him. Indeed, this learning can help to anticipate all possible situations that a human might find himself in and/or that time can change it. This learning process is defined as the key to increase the knowledge of any task after real observations. The first known work in this field was done by *Arthur Samuel* in 1952; he used the game of checkers to creat the first learning program. Many researchers like *Ross Quinlan, Thomas Dietterich, Gerald DeJong, Trevor Hastie, Robert Tibshirani, John Shawe-Taylor, Jerome H. Friedman, Vladimir, N. Vapnik* and others were intersted on studying the learning field which accelerate the progress of this field. Nowadays, a field in the Artificial intelligence (AI), called **Machine Learning**, is created to study, analyse and test all type of learning problem. And, the learning ability is provided by two main types of learning algorithms, the supervised and the unsupervised learning.

In the unsupervised learning, also called clustering methods [100], we try to discover the structure of the database. This usually means, we try to find groups, define clusters and construct classes using known informations from the database. The goal of clustering is descriptive.

In the supervised learning, each individual (input) in the database comes with a pre-assigned class label (output). The task is to train a function, called classifier, that maps from input to output and can labeled very well all the inputs of the database.

Then, this classifier is used to classify a new unclassified individual. Thus, the goal of supervised learning is predictive. One of the very known form of the supervised learning is the **Classification**, known also as pattern recognition or discrimination. The main particularity of the classification is that the outputs are usually categorical attributes.

In this thesis, we are intersted on studying the **supervised learning approaches**, especially the classification problem. Therefore, we need to construct two different samples from the original observed database. The first sample, called **training or learning sample**, is used to learn from its observation. This sample has a very important role in the learning process. It collects information of the observed data like find relations between or among variables or find how each variables progress with time... The second sample, called **test sample**, is used to prove and assess the strength and utility of informations collected from the training sample.

In this chapter, we are focusing on the supervised learning methods that try to solve the Classification problem. In the following, we describe a various learning methods which consist the main background theory of this Thesis like Ensemble Methods, Random Forests and Support vector machine.

2.1.1 Notations

Consider the task of a classification problem as follow:

Given a data set Ω of N individuals $\omega_i, i = 1, \ldots, N$ represented by p attributes $X^j, j = 1, \ldots, p$ in a representing space $\Re^p$ (inputs), and a membership class $Y \in \{y1, \ldots, y_k\}$ (outputs). Let X be the function mapping an individual to its representation and Y the function mapping to the class:

$$\begin{array}{lcccl} X & : & \Omega & \longrightarrow & \Re^p \\ & & \omega_i & \longrightarrow & X(\omega_i) = X_i = \left(X^j(\omega_i)\right)_{j=1,\ldots,p} \\ Y & : & \Omega & \longrightarrow & \mathcal{Y} = \{y_1, \ldots, y_k\} \\ & & \omega_i & \longrightarrow & Y(\omega_i) = Y_i = Y \end{array}$$

The set of inputs is called also instances, cases, observations or features, and $\mathcal{Y}$ is the set of class labels called also targets or outputs. If $\mathcal{Y}$ is a set of finite values (Ordinal or nominale), the learning problem is called Classification and Binary Classification when $\mathcal{Y}$ contains only two values. If $\mathcal{Y}$ is quantitative, the learning problem called

Regression. We will refer to an individual ω or to a point $X(\omega)$ with the same meaning, indistinguishably.

Each Y_i is generated by an unkown function f where $Y_i = f(X_i)$. We try to find a function h, an hypothesis, that approximates the true function f. The learning process consists to search through the space of possible hypotheses to find the only one hypothesis h (noted in the following sections ϕ) that classify well the training set. We suppose that the data set Ω is partitionned into two sets, a learning set E_l and a test set E_t, the E_l is formed of n individuals $(\omega_i)_{i=1,...,n}$.

To prove the efficiency of a learning method, we measure the accuracy of the hypothesis h. To do so, we use a new data, in this case it is the test set E_t (different from the learning set E_l). Accuracy (see Appendix A.1) is measured based on how well h predict the output of E_t. h generalises well if the Accuracy of h is high. The **Generalization** is a very important concept in machine learning. It characterizes the efficiency of how much results, found from a given training set, can be used to classify any new unclassified example. The ability of Generalization determines the efficiency of the learning algorithm.

We use the toy example of Table 2.1 to illustrate different definitions. This is a two classes data set Ω', subset of an Ω, formed of 17 individuals mapped into a two-dimensional space R^2. In the figure 2.1, the class $y_1 = 1$ is indicated by a bold dot and the class $y_2 = 2$ by an empty dot.

Ω'	X_1	X_2	Y	Ω'	X_1	X_2	Y
ω_1	2.13	2.33	2	ω_{10}	0	2.33	2
ω_2	2.13	4.11	2	ω_{11}	5.64	5.17	2
ω_3	2.22	1.76	2	ω_{12}	7.87	2.33	1
ω_4	3.37	6.88	1	ω_{13}	5.64	7.5	1
ω_5	6.77	0.67	1	ω_{14}	4.53	8.1	1
ω_6	4.53	1.16	1	ω_{15}	3.37	4.31	1
ω_7	3.37	0	1	ω_{16}	5.64	4.11	2
ω_8	1.8	6.47	2	ω_{17}	7.87	4.11	2
ω_9	0	5.77	2				

TABLE 2.1: Set of points in R^2 with two classes

2.1.2 Classifier

A wide variety of methods could provide individual classifiers. All machine learning methods can be used to generate a set of classifiers like decision trees, in particular the Random Forests [13] (see section 2.4), neural networks [112, 164], linear discrimination methods [136], support vector machine (see section 2.5) and the k-Nearest Neighbors

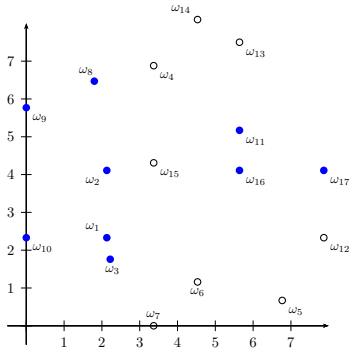

FIGURE 2.1: Scatter plot of Ω' in $\mathbb{R}^2$ with two classes

methods [59] (see section 2.3). Indeed, there exist the non classical framework methods such as stochastic discrimination [81, 82] and the methods based on projections in random sub-spaces [45, 60, 61].

The goal of any machine learning algorithm is to lead to a classifier capable of predicting, with high Accuracy, the membership class $Y(\omega)$ for any individual ω knowing its attribute values of $X(\omega)$. Basically, the prediction is based on the knowledge we can get about the probability distribution from the training data set:

$$P(Y/X) \quad = \quad (p(Y = y_l/X); l = 1, \ldots, k).$$

Generally, a classifier ϕ helps, somehow, for all individuals ω, to estimate $\hat{P}$ which is the membership probability vector for all classes. Thanks to the learning sample E_l, the predicted membership class is $\hat{y}_l$ the most likely, determined as follows:

$$\hat{y}_l \quad \text{is such that} \quad \hat{p}(Y = \hat{y}_l/X) = \max_{l=1,\ldots,k} \hat{p}(Y = y_l/X)$$

By thresholding at the maximum value of this vector, the membership class can be represented by a zero vector except for the most likely class by a value 1 at the corresponding rank: $\hat{P} = (0, \ldots, 0, 1, 0, \ldots, 0)$. If the classifier ϕ is considered as being quite reliable, then predicting the membership class for an individual ω is $\phi(X(\omega)) \in \{y_1, \ldots, y_k\}$.

We can define a classifier ϕ formally as follow:

Definition 2.1 (Classifier). In the representation space $\Re$ of p dimension, a classifier is an application from $\Re$ to the simplex of order l where $l \in \{1, \ldots, k\}$, k is the number of classes and a simplex of order l is

$$\mathcal{S}_l \quad = \quad \left\{ (\int_1, \ldots, \int_k) \in [0,1]^k s.t. \sum_{l=1}^{k} \int_l = 1 \right\}$$

So,

$$\begin{array}{rccl} \phi & : & \Re^p & \longrightarrow & \mathcal{S}_l \\ & & X(\omega) & \longrightarrow & \phi(X(\omega)) = (p_1, p_2, \ldots, p_k) \end{array}$$

Following, we describe many different classification methods in Machine Learning. The goal of these methods is to build a classifier that can learn efficiently from training set.

2.2 Ensemble Methods

Ensemble methods (EM) is a well known group of machine learning algorithms [165]. EM train multiple classifiers that solve the same original task then, combine them to find the final optimal classifier (see figure 2.2 page 8). These methods are more accurate and reliable than any of the single classifier in the ensemble [33]. Dietterich [31] gave three reasons of why good ensembles might be much better than a single classifier. Assuming that a machine learning algorithm search a space of possible hypothesis h that approximates an unkown fonction f, here are the three reasons:

1. The training set might not provide sufficient information for choosing a single best classifier.
2. The learning algorithm might not be able to solve the difficult search problems.
3. The hypothesis space might not contain the true fonction f but several equally good approximations to f.

As described previously, the dataset Ω is composed of two sets E_l the learning set and E_t the test set. An Ensemble Method consists of two steps:

1. Generate multiple diverse classifiers $\phi^1, \ldots, \phi^m$ by using different learning sets $E_l^1, \ldots, E_l^m$ subsets of the original learning set E_l
2. Combine these classifiers to construct the final optimal classifier ϕ

Generally, EM have good performance when classifiers in the ensemble is accurate and diverse. In this case, EM achieve strong generalization ability to exploit the unclassified new data. As mentionned in [165] and [31], the generalization ability of an Ensemble Method is usually better than that of a single classifier.

The main idea of EM is based on the ability of boosting weak classifier (original single classifier) by combining multiple classifiers, which is better than a random guessing of

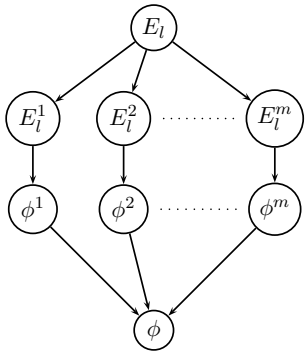

FIGURE 2.2: The EM general algorithm

the strong classifier. There are four main factors, described in [100], that explain the various ensembles methods:

- Relationship among classifiers.
- Methods of combining the classifiers.
- Diversity between classifiers.
- Number of classifiers in the ensemble.

Two popular methods of EM, Boosting [46] and Bagging [9], are known for creating the best accurate ensembles [166].

2.2.1 Bagging

Bagging [53], introduced by Breiman [9], is the first effective method of Ensemble Learning. It combines hypothesis learned from multiple Bootstrap [1] samples. Thus, its name is derived from "Bootstrap Aggregating". The Ensemble method, Bagging, try to improve machine learning algorithms. This method train classifiers generated by many different bootstrap samples. m Bootstrap samples $B_1, \ldots, B_m$ are produced. Each one of these bootstrap samples is used to build a classifier ϕ_i where $i = 1, \ldots, m$. Then, all the m classifiers are combined using the majority vote to find the final classifier ϕ (see algorithm 1 page 9). "Bagging predictors is a method for generating multiple versions of a predictor and using these to get an aggregated predictor" is the definition of Bagging writing by Breiman [9].

[1] Bootstrap [40] is a resampling method based on building samples randomly with replacement and these samples have all the same size of the original training set.

Since the original learning set is sampled with replacement, some instances are repeated and some are not present at all in the bootstrap sample. This Bootstrapping concept is provided to build many different classifiers, used when the classification method is unstable like the decision trees. Bagging improves the performance of unstable algorithms, in other hand, it can decrease the performance of stable method like k-nearest neighbor [9].

Algorithm 1 Bagging Algorithm

Require: Training set E_l with class label $Y(E_l)$, classification method F and m bootstrap samples
For $i = 1, \ldots, m$

1) Creat a bootstrap sample from E_l, B_i
2) Build a classifier $\phi_i = F(B_i)$

$\phi(x) = argmax_{y \in Y} \sum_{i=1}^{m} I(\phi_i(x) = y)$
where I is an indicator function that returns 1 if $\phi_i(x) = y$ is true and 0 otherwise

Output the classifier ϕ

2.2.2 Boosting

Boosting [53, 166] is the most used Ensemble method. Like any Ensemble method, it try to improve the performance of machine learning algorithms. This method works as an iterative algorithm that combine all classifiers into a single one. In each iteration, it aggregates weak classifier with other classifiers in order to find at the end a strong classifier that achieve higher accuracy than the weak classifier .

The Boosting idea was originated from the interseting question posed by Kearns and Valiant [75] of whether a weak learning algorithm which performs just slightly better than random guessing can be boosted or enhanced into an accurate strong learning algorithm. Schapire [130] in 1990 came up with the first provable boosting algorithm. Boosting improve the performance because of two main reasons, the final classifier has a small error on the training set and lower variance than those produced by the weak classifier. On the other hand, Boosting can destroy the generalization performance which causes an **overfitting** [2].

To understand how a boosting algorithm works, a simple example is illustrated in figure 2.3 page 10. In this figure, each circle corresponds to an example of the learning set; the size of the circle corresponds to the weight of each example; and, the black circle corresponds to the correctly classified example by the used hypothesis. The boosting

[2] It is the case when the learning result performs better on training set than on unclassified new data.

algorithm is based on a weighted learning set where each example is associated to a weight $w_j \geq 0$. More the weight of an example is high, more the importance attribute to this example during the learning process is high. We describe in the following paragraph the most common implementation of Boosting, AdaBoost.

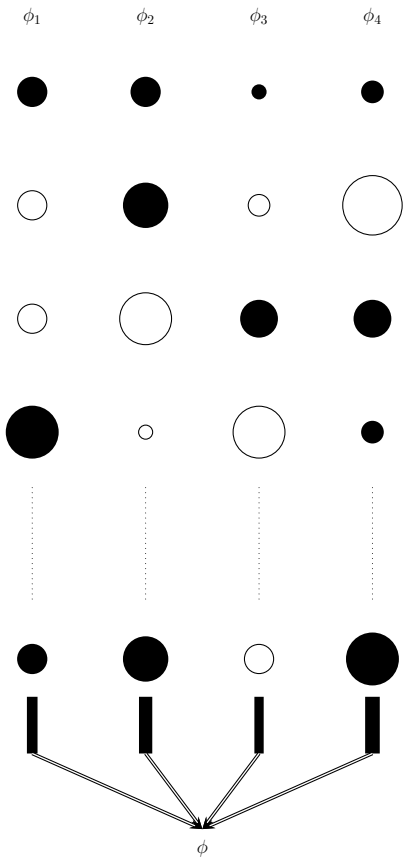

FIGURE 2.3: Simple example that illustrate the Boosting Algorithm

2.2.3 AdaBoost

In 1995, AdaBoost [46, 47] was introduced by Freund and Schapire that solve many problems of the earlier boosting algorithms. The AdaBoost algorithm is based on a very important property: Using a weak learning algorithm, Adaboost estimates the robust hypothesis that classifies mostly all the training set. A weak learning algorithm provides an hypothesis that has a slightly better accuracy than random guessing. Therfore, AdaBoost known as the most important algorithm among the Boosting algorithms.

Consider the case of binary classification where E_l is the learning set and $\mathcal{Y} = \{-1, +1\}$ the two class label. AdaBoost is an iterative algorithm. It try to construct a strong classifier as linear combination of simple weak classifiers. The idea of AdaBoost is to assign weights over the training set. The weight of the ith input, on the round j where $j = 1, \ldots, m$, is denoted $w_j(i)$. The process starts by attributing to all the examples

an equal weight. Then in every round, the weights of misclassified input are increased while the weights of the correctly classified input are decreased. Therefore, the weak learner is forced to focus on the hard inputs in the training set. AdaBoost is described in algorithm 2. Let a weak classifier be ϕ_j, the goodness of ϕ_j is measured by its error:

$$\epsilon_j = P[\phi_j(X_i) \neq Y_i] = \sum_{i:\phi_j(X_i)\neq Y_i} w_j(i)$$

The classification of a new input $\mathbf{x_{new}}$ from E_t is made by voting on all classifier ϕ written as follow:

$$\phi(\mathbf{x_{new}}) = sign\left(\sum_{j=1}^{m} \alpha_j \phi_j(\mathbf{x_{new}})\right)$$

where α_j are the new weights found by AdaBoost.

Algorithm 2 AdaBoost Algorithm

Require: Learning set E_l, $\mathcal{Y} = \{-1, +1\}$ and $\mathbf{x_{new}} \in E_t$

Initialize $w_1(i) = 1/n$
For $j = 1, \ldots, m$

1) Train weak classifier using w_j distribution
2) Get weak classifier ϕ_j with error $\epsilon_j = P[\phi_j(X_i) \neq Y_i]$
if $\epsilon_j > \frac{1}{2}$ **Stop**
3) Choose $\alpha_j = \frac{1}{2} log(\frac{1-\epsilon_j}{\epsilon_j})$
4) Set new weights: $w_{j+1}(i) = \frac{w_j(i)\ exp(-\alpha_j Y_i \phi_j(X_i))}{Z_j}$
where Z_j is a normalization factor which enables w_{j+1} to be a distribution

Output the final classifier:
$\phi(\mathbf{x_{new}}) = sign\left(\sum_{j=1}^{m} \alpha_j \phi_j(\mathbf{x_{new}})\right)$

2.3 K Nearest Neighbors

KNN, an Instance Based Learning (IBL[3]) classifier, is the simplest classifier in machine learning used usually as a benchmark classifier [2, 22, 26, 41]. KNN method classify unlabeled examples based on their similarity with labeled examples in the training set. For an unlabeled example x_{new}, KNN method tries to find the k-"closest" labeled examples in the training set and then assign to x_{new} the class that appears most frequently

[3]IBL is a machine learning algorithm that classifies new examples comparing them to those already seen.

with the k-examples, which means that KNN method uses the majority vote among the k-closest labeled examples to determine the class label of x_{new}.

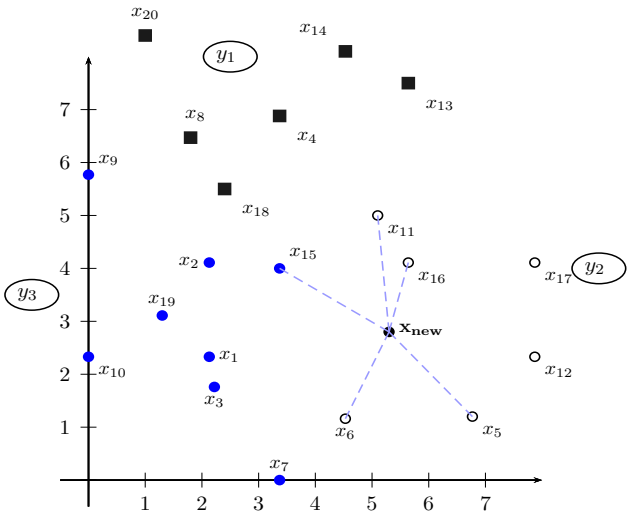

FIGURE 2.4: K nearest Neighbours (k=5)

To illustrate this method, a simple example formed of 20 inputs and 3 classes $Y = \{y_1, y_2, y_3\}$ is presented in figure 2.4 page 12. x_{new} is the unlabeled input that using KNN we try to find its class label. To do so, using $k = 5$, we find that among the five closest neighbors, four belong to y_2 and one belong to y_3. Thus, x_{new} is assigned to the predominant class y_2.

The performance of KNN is determined by the choice of :

1. an integer k
2. a metric to measure the proximity

Usually, the KNN method applies the Euclidean distance to measure the closeness between examples. Many works have been done to improve the KNN method [16, 93, 143] by changing or weighting the distance metric. Hastie and Tibshirani [57] developed an adaptive form of nearest neighbor classification (DANN) by using a local linear discriminant analysis to estimate an effective metric for computing neighborhoods. Weinberger and al [154] showed how to learn a mahalanobis distance metric for k-nearest neighbor classification by using semi-definite programming. This method, called large margin nearest neighbor (LMNN) classification, try to find a large margin that separates examples from different classes and keeps a close distance between the nearest neighbors of the same class labels. Furthermore, Song and al [137] introduced "Informativeness" as a new useful concept for distance metric in order to improve the KNN method. They define an example x_i of the training set to be "informative" if it is close to the unlabeled example and far away from the examples with different class labels.

The main problem of KNN method is the choice of k, the key of performant KNN. A smaller k means higher variance (less stable) and larger k means higher bias (less precise).

The use of large values of k provides smoothing that reduces the risk of overfitting due to noise in the data. However, too large value of k destroy the locality of the estimation of the class label because farther examples are taken into account and increases the computational costs. Therefore, k can be in units or tens rather than in hundreds or thousands.

kNN is very simple to understand and easy to implement, but, on other hand, it has some significant disadvantages. The kNN algorithm stores all the examples in the training set which create high storage requirements, computational cost and high susceptibility to the curse of dimensionality. As almost all the relevant information for the classification is located around the decision boundaries, some methods, called edited kNN, have been created to take advantage of the redendancy information of kNN algorithm.

- The first one, developed by Wilson in 1972 [157], is to classify all the examples in the training set and remove those examples that are misclassified in an attempt to separate classification regions by removing ambiguous points.
- The second one, developed by Riltter and all in 1975 [128], is the opposite of the first one. it attempt to remove training examples that are classified correctly in a purpose to define the boundaries between classes by eliminating points in the interior of the regions.

2.4 Learning Trees

2.4.1 Decision tree

Decision tree, or tree-based, method is considered to be one of the most popular classification technique used in various disciplines such as Machine Learning and pattern recognition. Decision tree is based on a very simple idea, build a tree. This tree is formed of edges and three types of nodes:

1. A root node: has no incoming edges
2. Internal nodes each one has one incoming edge and two or more outgoing edges; represented in figure 2.5 by a rectangle

3. Leaf or terminal nodes: each one has one incoming edge and no outgoing edges; represented in figure 2.5 by a circle

To illustrate the decision tree, we used the **playTennis** example, described in figure 2.5 page 14, where $Y = \{Yes, No\}$ is the set of the class labels, $X = \{Outlook, Humidity, Wind\}$ is the set of inputs. $X_1 = Outlook$ has 3 attributes (Sunny, Overcast, Rain), $X_2 = Humidity$ has 2 attributes (High, Normal) and $X_3 = Wind$ has 2 attributes (Strong, Weak).

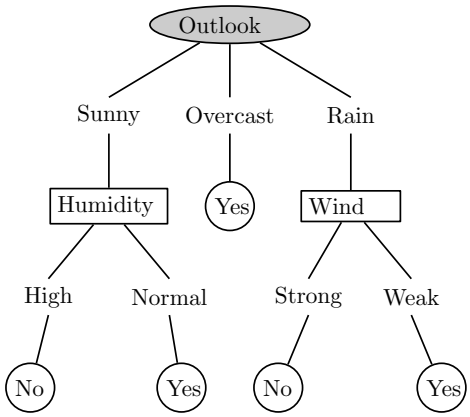

FIGURE 2.5: **playTennis** Example

In a decision tree, each internal node is splitted into two or more sub-branches according to a discrete function of the input attributes values. This discrete function is usually univariate which means based on the value of a single attribute. The decision tree is grown by recursive splitting until a stopping criteria is triggered. This criteria can be that tree reachs the maximum deepth or that all attributs of each input belong to a single class label. The leaf node assigns the classification results. New examples are classified by passing their inputs values from root of the tree down to a leaf node, according to the result of the tests along the path. Decision tree has several advantages [129]. It's simple and easy to understand. This tree is non parametric where nominal and numerical data can be used. Using this tree, decision rules are easy to interpret. It is known as a robust method with regard to outliers and missing values. On the other hand, this tree has some disadvantages [129] such as, over-sensitivity to the training set and to irrelevant attributes, low prediction accuracy and fair interpretibility of large trees.

In this context, many univariate criteria are used in the literature such as Information gain [23], Gini index [43], DKM criterion [30, 74], Gain ratio (see C4.5 in 2.4.2) and others [129]. There exist other stopping criteria [129]. Generally, using a rude or weak stopping criteria tends to under-fitted or over-fitted the decision tree. Therefore, pruning methods, suggested by Breiman and al in 1984 [15], were developed in order to improve the performance of a decision tree. Many techniques are used for pruning decision tree

such as Cost-Complexity Pruning (Breiman and al 1984 [15]), Ruduced Error Pruning (Quinlan 1987 [124]), Minimum Error Pruning (MEP) (Olaru and Wehenkel 2003 [111]).

Many efficient algorithms have been developed to construct the optimal decision tree like ID 3 (Quinlan 1986 [123]), C4.5 (Quinlan 1993 [125]), CART (Breiman and al 1984 [15]) and CHAID (Kass 1980 [72]).

2.4.2 C4.5

C4.5, a basic decision tree inducer, is a top down decision tree induction algorithm, developed by Ross Quinlan [84, 125]. It is an amelioration of the ID3 algorithm created also by Quinlan [123] to generate decision trees.

The C4.5 algorithm is used for classification and can handle numeric attributes. C4.5 tries to build a small and simple tree. Smaller trees are easy to understand and have higher predictive accuracy [126]. Since it is not easy to keep a decision tree minimal, C4.5 attempts on greedy search to select the optimal splitting function. C4.5 uses the Gain ration as splitting evaluation criterion, formulated as follow:

Let $F(y_j, \Omega)$ be the relative frequency of examples in Ω (the data set) that belong to class y_j where $j = 1, \ldots, k$. We consider a function A that partition Ω into t subsets $\Omega_1, \ldots, \Omega_t$.

$$GainRatio = Gain(\Omega, A)/split(\Omega, A)$$

where

$$split(\Omega, A) = -\sum_{i=1}^{t} \frac{|\Omega_i|}{|\Omega|} log(\frac{|\Omega_i|}{|\Omega|})$$

And

$$Gain(\Omega, A) = I(\Omega) - \sum_{i=1}^{t} \frac{|\Omega_i|}{|\Omega|} I(\Omega_i)$$

where

$$I(\Omega) = -\sum_{j=1}^{k} F(y_j, \Omega) log(F(y_j, \Omega))$$

$I(\Omega)$, called the Entropy of Ω, is the information needed to identify the class of an example of Ω.

Usually, the function that maximizes the $GainRatio$ is chosen to split the data set Ω which build up the decision tree.

C4.5 uses the "divide and conquer" algorithm [125] which partitions the training set until every leaf contains cases of a single classe. Consequently, the decision tree will

classify correctly all the training examples which will cause an overfitting. Generally, it leads to a loss of predictive Accuracy. To avoid Overfitting, C4.5 uses a stopping criterion. Indeed, To improve the results of c4.5, pruning the decision tree can be used which means removing some structure of the decision tree after it has been produced. The Error-based prunning method [129] is the method used by C4.5.

2.4.3 Random Forests

Random Forests (RF) is a powerful new approach proposed by Leo Breiman[4] [12] for classification and regression problem [24, 58]. it is one of the most successful Ensemble Methods (see EM in 2.2) based on trees.

RF uses a large number of unpruned decision trees created by a random split at each node of the decision tree. Normally, a tree with exact splits is more accurate than a tree created with radom splits. But, combining several of RF tree often is doing better than a single tree with exact splits. That is one of the main innovation of Random Forests. RF is characterized by three variants (see figure 2.6 in page 17):

1. The procedure used to generate the modified training sets used to construct trees: Bootstrap
2. The random way each tree is constructed: CART trees
3. The trees aggregations: Ensemble Method

Trees are grown using a binary partitionning. To split each node of each tree in RF, a randomly selected sample of r variables is used. The r, known as $mtry$ parameter, is usually $\sqrt{p}$ or $log2p$ where p is the number of all variables in the data set. Then, m Bootstrap samples are constructed in order to train each tree. The remainning inputs left out of the bootstrap samples, called Out-Of-Bag (OOB) are used to estimate the error rate and the variable importance. In RF, a proximity measure is defined as the number of time a unique observation of the training set end up in the same leaf node. This definition is Based on the idea that more two example are similar more they will land in the same leaf node of a tree. This proximity measure is used to fill in the missing value and to detect the outliers. To assign a class to new observation, we apply the majority votes using all the trees.

An new important feature in RF is the OOB error. It is considered as an internal validation for the constructed trees. The OOB error is the average error frequency

[4]Leo Breiman: co-author of CART with Friedman, Olshen and Stone [15] and author of Bagging [9] and Arcing [10] approaches to combime trees.

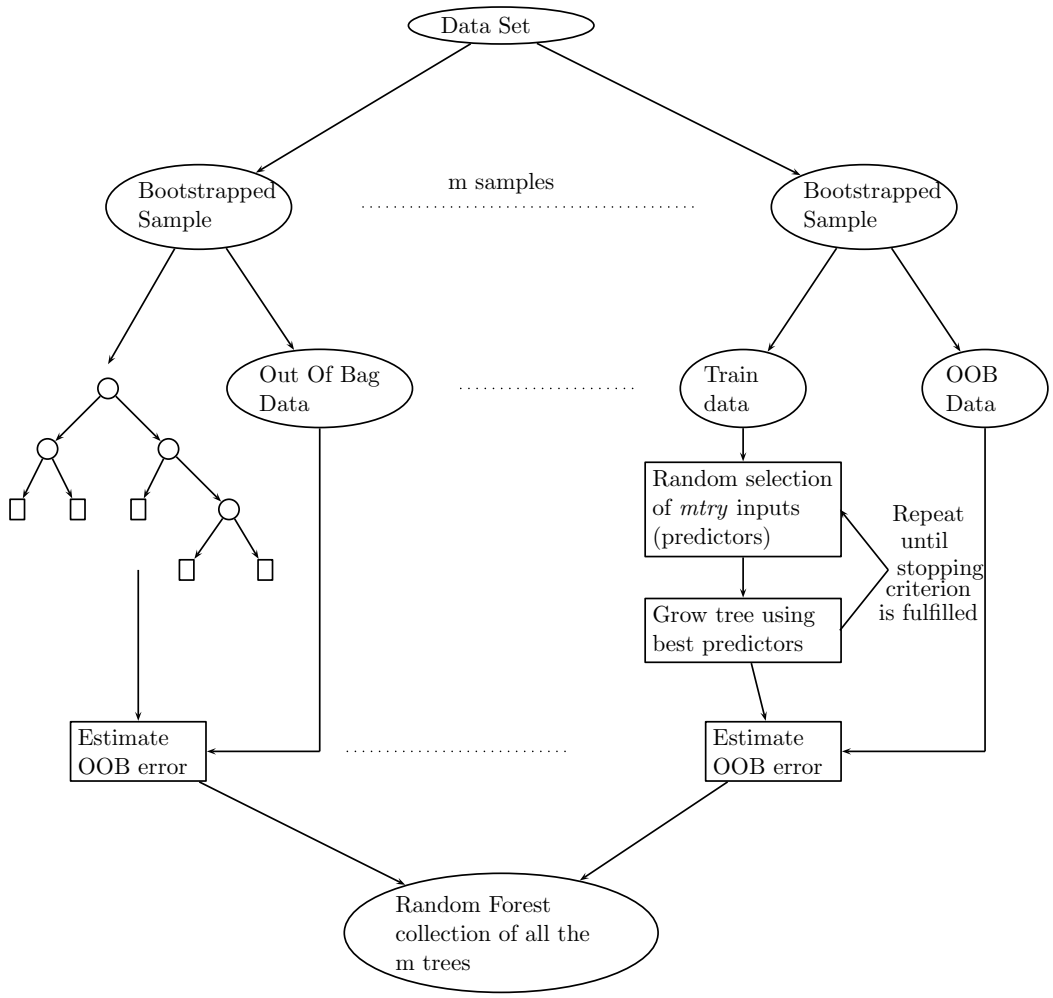

FIGURE 2.6: The Random Forests Procedure

obtained by applying each three to the OOB data. This feature is used to improve the result of RF.

Random Forests has become a major tool for classification problem because of it has many advantages [14] such as:

- High levels of predictive accuracy

Algorithm 3 The algorithm of Random Forest

Require: Data set Ω of size N, p variables, m the number of bootstrap samples and Stp stopping criterion

For $b = 1$ to m
1) Draw a bootstrap sample Ω^b from Ω
2) Grow a Random Forests tree T_b using Ω_b by repeating the following steps for each leaf node, until a Stp is reached :
a) Select randomly r variables
b) Pick the best spliting variable among the r variables and then use it to split the tree
Output the ensemble of trees $(T_b)_1^m$

To predict the class of a new point $\mathbf{x_{new}}$

Let $\phi_b(\mathbf{x_{new}})$ be the class prediction of the bth Random Forests tree.
$\phi_{RF}(\mathbf{x_{new}}) = majorityVote\{\phi_1(\mathbf{x_{new}}), \ldots, \phi_b(\mathbf{x_{new}}), \ldots, \phi_m(\mathbf{x_{new}})\}$

- Train rapidly even with many of potential predictors
- Resistant to overfitting
- Relatively robut to outliers and noise
- Manage the misssing data
- Self testing using the OOB data
- Give variables importance

2.5 Support Vector Machine

Among all the kernel methods, inspired from learning of Vapnik, the Support Vector Machine (SVM) is the most known in resolving the problem of binary pattern recognition or binary classification [18, 63, 131]. SVM is a very efficient technique for classifiying data. It attempts to find an optimal separation between two classes using a linear classifier in an appropriate space. The main reasons that make SVM attractive are :

- SVM constructs the optimal margin separator that provide a good generalization
- SVM creates a linear separating hyperplane even if the training set is not linearly separable in the original space. Using the **kernel trick**, An hyperplane is transformed to be linearly separable in higher dimensional space.

For instance, we are using the case of the binary classification where $\mathcal{Y}$ is $\{-1, +1\}$ We try to predict the outputs y for unseen inputs $x_{new} = X(\omega_{new})$. To study the performance of SVM method, we measure a loss function on the outputs that state how bad is the predicted $\hat{y}$ to the true one.

Following, we describe the three main cases very known in SVM, the linear, the non linear and the soft SVMs. The linear seperator case develops all the basis of the SVM algorithm. The non linear case represents the innovation and the successful part of SVM. Then, the soft case represents as an adjustment of the previous two cases . Let us started by the definition and the advantage of a kernel function.

2.5.1 Kernel

To study the problem of binary classification, we try to predict the output $y \in \{-1, +1\}$ of a new instance $x' \in \Re^p$. Which mean informally, that we choose this y such that (x', y) is similar on some way to the training examples in Ω.

To formalize this, we have to define the similarity in $\Re^p \times \{-1, +1\}$. Obviously, the similarity of the outputs in $\{-1, +1\}$ is one of two situations either identical or different. On the other hand, the choice of similarity measure for inputs, a critical subject on machine learning, is described as following:

Let us use a similarity measure $S : \Re^p * \Re^p \longrightarrow \Re$

$$(x, x') \longrightarrow S(x, x')$$

We assume that S is symmetric; $S(x, x') = S(x', x)$ for all $x, x' \in \Re^p$. we can use a dot product or scalar product as a similarity measure because it deals with the instances geometrically and let it easy to use linear algebra and analytic geometry.

Inputs can be any kind of object. Therfore, we do not know if these inputs are in a inner product space. Thus, in some case, we can represent the inputs x' as vectors in a inner product space noted $\mathcal{H}$ using a Φ map as follow:

$$\Phi : \Re^p \longrightarrow \mathcal{H}$$

$$x' \longrightarrow \Phi(x') = \mathbf{x}'$$

where $\mathbf{x}'$ *is a vector.*

Then, $S(x, x') =< \mathbf{x}, \mathbf{x}' >=< \Phi(x), \Phi(x') >$ where $<, >$ is the canonical dot product. This similarity measure is known as a kernel function noted $k(x, x')$ and the $\mathcal{H}$ space called a feature space.

The non linear case where we cannot find a linear seperator between the classes, represents a serious problem in support vector machine. Here a simple example that describe this case:

Let $(0,0),(0,1),(1,0)$ *and* $(1,1)$ be the training set, where $(0,0)$ and $(1,1)$ belong to the same class label y_1 and $(0,1)$ and $(1,0)$ belong to the same class label y_2 (see figure 2.7 in page 20).

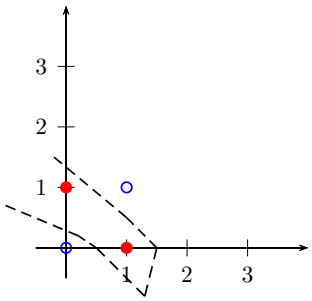

FIGURE 2.7: Non linear separator

In this situation, we can use the kernel trick where the non linear classifier in the input space can be a linear classifier in some high dimensional space, the feature space $\mathcal{H}$ (see non linear SVM in subsection 2.5.3). Moreover, using kernels, all the SVM computations of a linear classifier in the feature space are performed directly in the input space. For the example described above, we use k :

$$\begin{array}{rcl} \Re^2 & \longrightarrow & \Re^3 \\ (x,y) & \longrightarrow & k(x,y) = (x,y,x*y) \end{array}$$

The kernel function k, illustrates in figure 2.8, transforms the training set into $(0,0,0)$, $(0,1,0)$, $(1,0,0)$, $(1,1,1)$ which are linearly separable.

2.5.2 Linear SVM

Optimal Hyperplane

For a typical two class classification problem, Vapnik and al. [131] considered that an hyperplane (see figure 2.9 in page 22), in a inner product space $\mathcal{H}$, which separates the two classes (y_1, y_2), is:

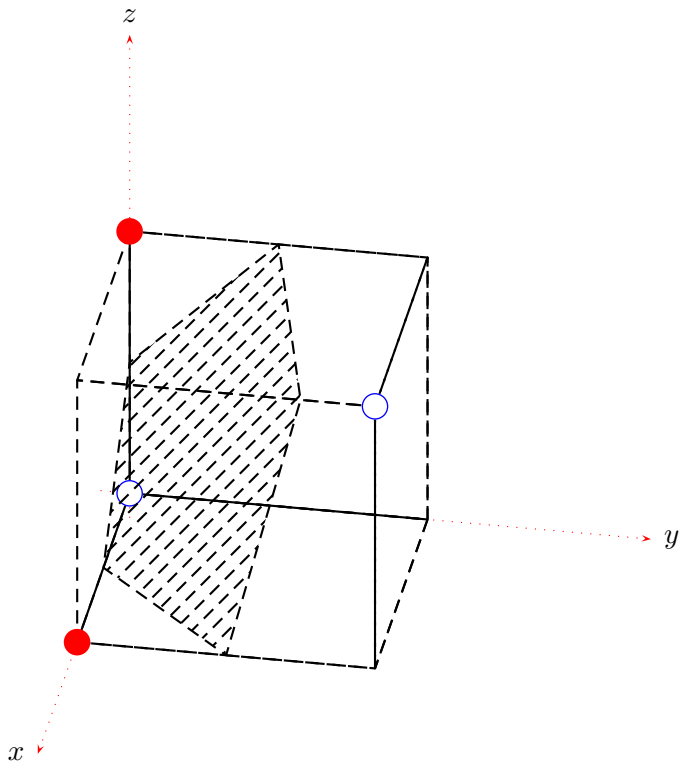

FIGURE 2.8: linear separator using the kernel k

$$< w, x > +b = 0 \tag{2.1}$$

where $w \in \Re^p$ and $b \in \Re$ corresponding to the decision function,

$$f(x) = sgn(< w, x > +b) \tag{2.2}$$

w is normal to the hyperplane described in Equ 2.1. $b/\|w\|$ is the perpendicular distance from the hyperplane to the origin where $\|w\|$ is the euclidean norm of w.

We define the **margin** of a separating hyperplane by $margin \ = d_+ + d_-$ where d_+ (d_-) is the shortest distance from the separating hyperplane to the closest element of y_1 (y_2).

In the linear separable case, the support vector machine algorithm try to find the separating hyperplane with the largest margin, called **Optimal Margin**.

The data points, in a linear separable case, will be classified as follows:

$$< \mathbf{w}, x_i > +b \geq +1 \qquad \text{for } y_i = +1 \tag{2.3}$$

$$< \mathbf{w}, x_i > +b \leq -1 \qquad \text{for } y_i = -1 \tag{2.4}$$

Equ 2.3 and Equ 2.4 can be combined into one set of inequalities :

$$y_i(< \mathbf{w}, x_i > +b) - 1 \geq 0 \qquad \forall i \tag{2.5}$$

$\frac{|1-b|}{\|\mathbf{w}\|}$ is the perpendicular distance from H_1, the positive hyperplane : $< \mathbf{w}, x_i > +b = +1$, to the origin. Then, the distance between the hyperplane described in Equ 2.1 and H_1 is $\frac{1}{\|\mathbf{w}\|} = d_+$. For the negative hyperplane H_2 : $< \mathbf{w}, x_i > +b = -1$, $\frac{|-1-b|}{\|\mathbf{w}\|}$ is the perpendicular distance from H_2 to the origin. And then, $\frac{1}{\|\mathbf{w}\|} = d_-$.

Hence, the **margin** is $\frac{2}{\|\mathbf{w}\|}$ ($d_+ = d_- = \frac{1}{\|\mathbf{w}\|}$).

Moreover, as H_1 and H_2 are parallel (same normal $\mathbf{w}$) and no points fall between them, we use this two hyperplanes to maximize the **margin** by minimizing $\|\mathbf{w}\|^2$. Therefore, the SVM find an optimal separating hyperplane with maximum margin by solving the following optimization problem, called the **primal optimization problem**:

$$\min_{\mathbf{w} \in \mathcal{H}, b \in \Re} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to : } \; y_i(< \mathbf{w}, x_i > +b) - 1 \geq 0 \;\; \forall i \tag{2.6}$$

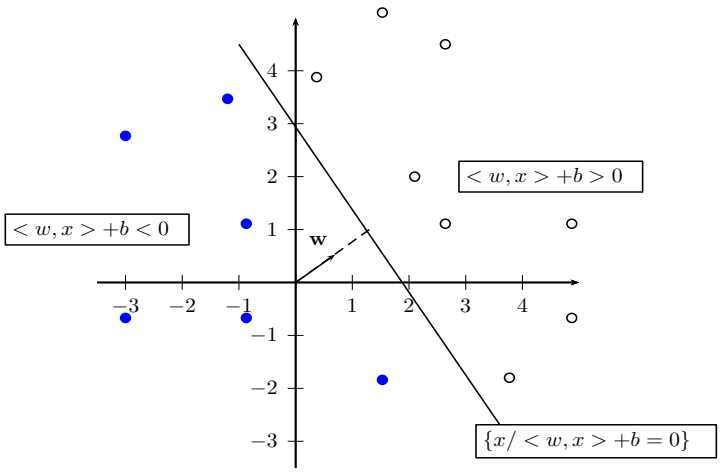

FIGURE 2.9: Representation of the Data set

For a typical two dimensional space, the solution will have the form shown in figure 2.10 page 23. These points that lie on one of the hyperplanes H_1, H_2 (verifiying the inequalities in Equ 2.5), called **support vectors**, play a fundamental role in SVM algorithm.

To solve the primal optimization problem in Equ 2.6, it is more convenient to use **Lagrange** function:

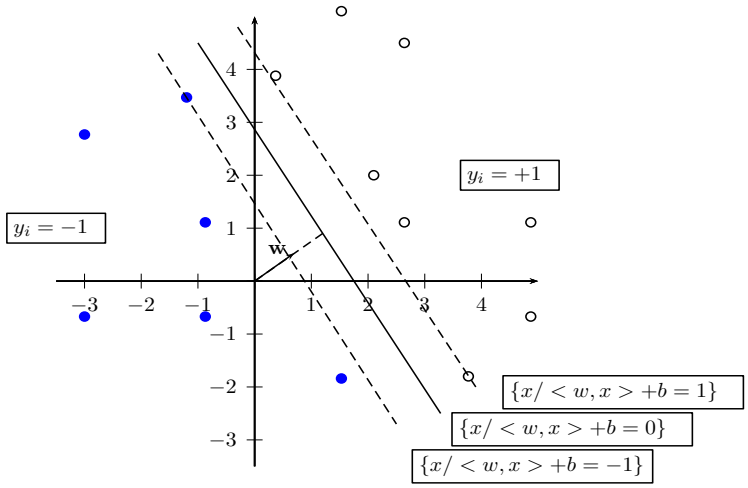

FIGURE 2.10: Optimal hyperplane

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(< \mathbf{w}, x_i > +b) - 1) \tag{2.7}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$.

The Lagrangian L must be minimized with respect to the primal variables $\mathbf{w}$ and b and maximized with respect to the non negative dual variable $\boldsymbol{\alpha}$. Consequently, the derivatives of L with respect to $\mathbf{w}$ and b must be:

$$\frac{\partial L}{\partial b}(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \Longrightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.8}$$

$$\frac{\partial L}{\partial \mathbf{w}}(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \Longrightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i} \tag{2.9}$$

Then, we can use the **Karush-Kuhn-Tucker (KKT) conditions** for the optimization constrained in Equ 2.7. It is sufficient and formally described as follow:

$$\alpha_i [y_i(< \mathbf{w}, x_i > +b) - 1] \geq 0 \qquad \forall i \tag{2.10}$$

Notice that there is a lagrange multiplier α_i for every input x_i. According to KKT conditions (Equ 2.10), the inputs x_i for which $\alpha_i > 0$, called **support vectors**, lie exactly on the margin. All other inputs that have $\alpha_i = 0$, are irrelevant, they satisfy automatically Equ 2.5 and do not appear in Equ 2.9.

To solve the problem, we substitute Equ 2.8 and Equ 2.9 into the Lagrangian in Equ 2.7. We find the **dual form** of the optimization problem :

$$\max W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j < x_i, x_j > \tag{2.11}$$

subject to : $\alpha_i \geq 0, i = 1, \ldots, n$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$

The **dual optimization** problem can be solved using a standard optimization algorithm to find α_i. As an immediate application, $\mathbf{w}$ is explicitly determined by the training procedure using Equ 2.9. b is easily found by using KKT condition Equ 2.10 and the **support vectors** $\alpha_i \neq 0$ which means $y_i(< \mathbf{w}, x_i > +b) = 1$.

Then, we obtain an **optimal hyperplane** by substituting the result of Equ 2.9 in the equation 2.2 of the decision function :

$$f(x) = \sum_{i=1}^{n} y_i \alpha_i < x_i, x > +b \tag{2.12}$$

and an indicator decision function $sgn\ f(x)$

So, we simply determine on which side of the decision boundary a given input x lies and assign the corresponding class label. The class of x is the sign of $f(x)$.

We can write one expression for the Optimal hyperplane as an indicator decision function as follow:

$$f(x) = sgn\left(\sum_{i=1}^{n} y_i \alpha_i < x_i, x > +b\right) \tag{2.13}$$

To conclude, we evaluate this decision function in terms of dot product taken between the inputs to be classified and the **support vectors**.

2.5.3 Non linear SVM

If the decision function is not a linear function, the SVM proposes to transform the input data space into a higher dimensional feature space (Hilbert space $\mathcal{H}$) via a mapping Φ function where we can apply the steps described previously in the linear case. Φ is defined as follows:

$$\begin{array}{rccl} \Phi & : & \Re^p & \longrightarrow & \mathcal{H} \\ & & x & \longrightarrow & \Phi(x) \end{array}$$

In this case, we can write the optimization problem:

$$MaxW(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - 1/2 \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j < \Phi(x_i), \Phi(x_j) > \tag{2.14}$$

subject to :

$$0 \leq \alpha_i \leq C, i = 1, \ldots, n \tag{2.15}$$

and

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.16}$$

and the solution is :

$$f(x) = \sum_{i=1}^{n} y_i \alpha_i < \Phi(x_i), \Phi(x) > + b \tag{2.17}$$

We find that the problem and the solution, in the non linear case, depend on the dot product $< \Phi(x), \Phi(x') >$ in a high dimension space. So, to reduce these expensive calculations of dot product in the feature space, we use a positive definite kernel function K. Using K, we would never need to explicitly know or calculate Φ.

$$\begin{array}{rccl} K & : & \Re^p * \Re^p & \longrightarrow & \Re \\ & & (x, x') & \longrightarrow & K(x, x') =< \Phi(x), \Phi(x') > \end{array}$$

This kernel must satisfy the **Mercer's condition**. In general, the Mercer's condition is stated as a mathematical integral equation. This integral can be reduced to the following summation equation: for all $(x_i)_{i=1,\ldots,n} \in \Re^p$ and for all $(a_i)_{i=1,\ldots,n} \in \Re$, K is symmetric semidefinite positive if and only if

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

Also, Mercer's condition is fulfill if for K a symmetric kernel function, $(K(x_i, x_j))_{i,j}$ is a positive definite matrix [131]. In the litterature, there exist many known kernel function, some are shown in table 2.2.

	Linear	Polynomial	Gaussian	Radial basis function
Kernel $K(x, x')$	$x.x'$	$(c + x.x')^d$	$e^{\|x.x'\|^2/\sigma}$	$e^{-\gamma\|x.x'\|^2}$

TABLE 2.2: Examples of kernel functions

The optimization problem can be solved using the method for solving the optimisation in the separable case. Therefore, the decision function has the following form:

$$f(x) = sgn\left(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + b\right) \tag{2.18}$$

If we analyse the resolution process of this case, we can see that whatever the space of the inputs, there exist, in a high dimensional space $\mathcal{H}$, a linear seperator hyperplane using the SVM algorithm.This result is presented in a new different way based on kernel concept. Because it creates a seperator hyperplane that we cannot imagine it exists. But, the only problem of this result is the **Overfitting**. In the figure 2.11, we distinguish between the Linear separator hyperplane and the non linear hyperplane. We can see that the non linear case seperate exactly the two classes which causes the overfitting. On the other hand, the linear hyperplane can be a very good seperator of the two classes but the only problem is the misclassification of some individuals. To resolve this problem, we use the soft margin case which can be seen as an adjustment of the linear and non linear case.

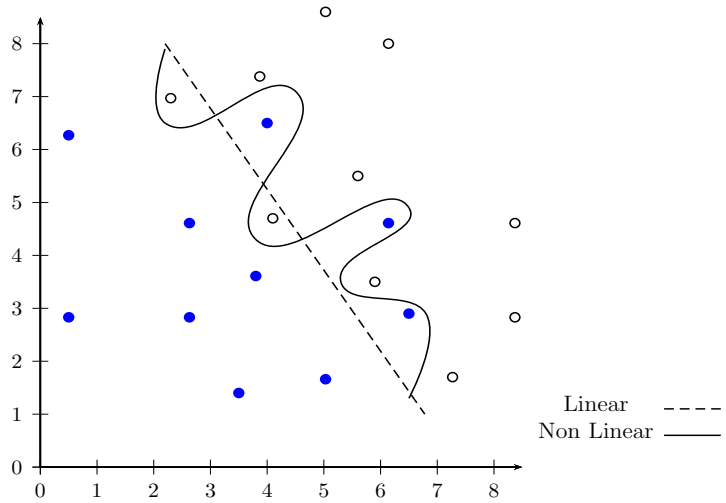

FIGURE 2.11: Two seperators: linear and non linear hyperplane

To illustrate the Overfitting problem of a non linear SVM, we used a different example (see appendix A.2) more visually understandable. In figure 2.12, we apply the KSVM algorithm with classification type C-svc using the R `kernlab` library to this example. In each graph of figure 2.12, we changed the C parameter [5] of the KSVM algorithm: C= 1, 10, 100 and 1000. We can see that by increasing C, the classification tend to be perfect causing the overfitting.

[5]Cost of constraints violation by default is equal to 1: this is the C-constant of the regularization term in the Lagrange formulation.

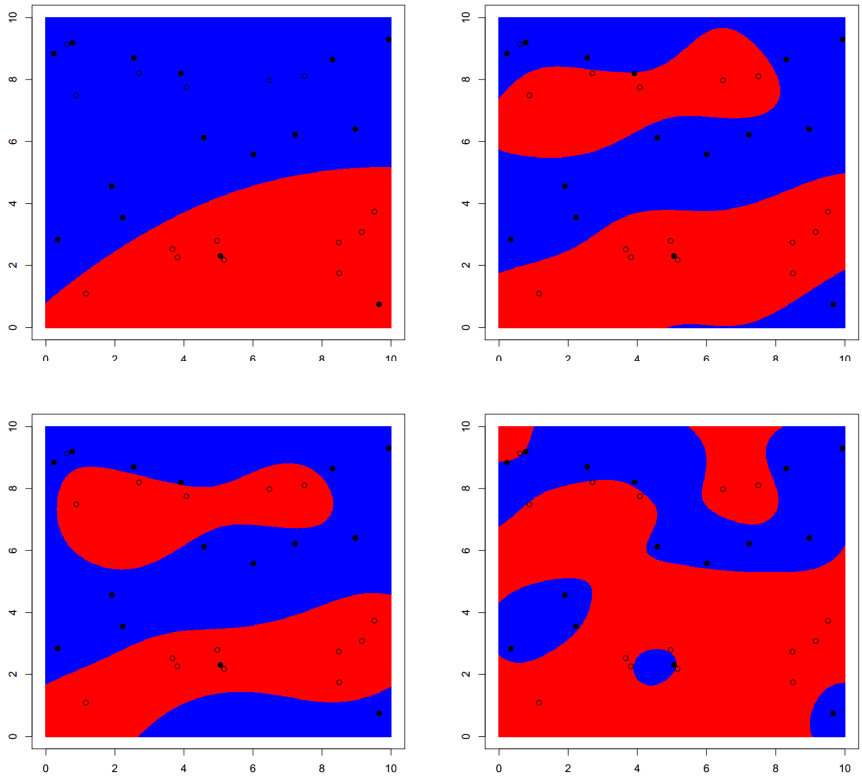

FIGURE 2.12: Overfitting caused by a non linear hyperplane: *C = 1 (up left), C=10 (up right), C=100 (down left) and C=1000 (down right)*

2.5.4 Soft Margin Hyperplane

In practice, the seperable linear case do not exist for a real data, and if it exist, the data may contain an individual outlier or many outliers. which means, some points (outliers) are not in the right side of the hyperplane (see figure 2.13 in page 28). And, that can affect the separating hyperplane found in the linear case described above. Moreover, the non linear case presents overfitting problem. To solve all these problems, we use an approach created by Cortes and Vapnick [21] based on adding the so-called **slack variables** $\xi_i \geq 0, i = 1, \ldots, n$ where ξ_i is the distance between the outlier and his right margin (see figure 2.13). To resume, we extend the algorithm of seperable data by adding ξ_i to make a relaxed separation constraints using Equ 2.3 and Equ 2.4 as follow:

$$< \mathbf{w}, x_i > +b \geq +1 - \xi_i \qquad for\ y_i = +1 \tag{2.19}$$

$$< \mathbf{w}, x_i > +b \leq -1 + \xi_i \qquad for\ y_i = -1 \tag{2.20}$$

Which combined to one separation constraints:

$$y_i(< \mathbf{w}, x_i > +b) \geq 1 - \xi_i \qquad \forall i \tag{2.21}$$

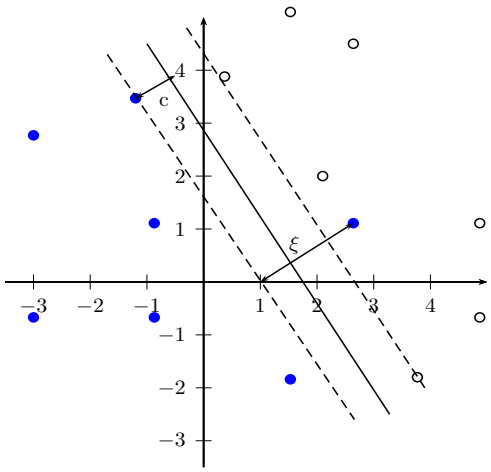

FIGURE 2.13: Soft margin hyperplane

Moreover, we can clearly see that, with a larger ξ_i (exceed the unity), the constraint in Equ 2.21 will be violated. To solve this problem, we penalize ξ_i by adding to the primal optimization problem in Equ 2.6 an extra cost for error comes from ξ_i , $C(\sum_{i=1}^{n} \xi_i)$ where a larger C assign a higher penalty to errors. Here is the formal expression of the textbf Primal Optimization problem:

$$\min_{\mathbf{w} \in \mathcal{H}, \xi \in \Re} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i \tag{2.22}$$

This model can be solved using Lagrangian method, the same procedure described in the previous paragraph. We find almost the same **dual Optimization** problem :

$$\max W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - 1/2 \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j < x_i, x_j > \tag{2.23}$$

subject to :

$$0 \leq \alpha_i \leq C, i = 1, \ldots, n \tag{2.24}$$

and

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.25}$$

The solution is again given by

$$\mathbf{w} = \sum_{i=1}^{s} \alpha_i y_i \mathbf{x_i} \tag{2.26}$$

and the decision function is :

$$f(x) = sgn\left(\sum_{i=1}^{s} y_i \alpha_i < x_i, x > +b\right) \tag{2.27}$$

where s is the number of **support vectors** ($\alpha_i \neq 0$). The only difference between the seperable case result and the non seperable case is that α_i have an upper bound C.

2.5.5 Multi-class SVM

The SVM is designed for binary classification. To extend it for multiclass classification, several methods have been proposed based on combining several binary classifiers (see figure 2.14 in page 29). The most known methods are "one-against-all" and "one-against-one".

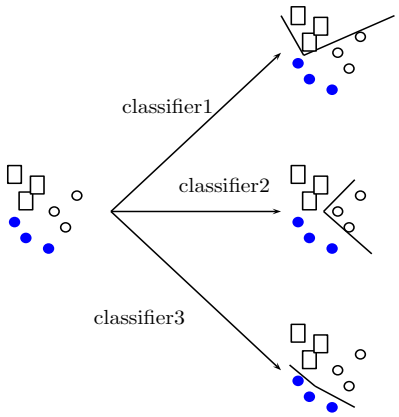

FIGURE 2.14: A simple example, Multi-class SVM

2.5.5.1 One-Against-All

In this method, we try to apply SVM for a multiclass classification by creating n different binary classifiers. To do so, we assume that, for the ith classifier, all the inputs in the ith class are positive and all remaining inputs are negative.

Let f_i be the ith classifier. To classify a new input x_{new}, we use :

$$f(x_{new}) = \arg\max_i f_i(x_{new}) \tag{2.28}$$

2.5.5.2 One-Against-One

In this method, To apply SVM for a multiclass classification, we build $n(n-1)/2$ different binary classifiers that separate between two classes i and j among the n classes.

Let f_ij be the classifier where all inputs of class i are positive and all inputs of class j are negative . To classify a new input x_{new}, we use :

$$f(x_{new}) = \arg\max_i(\sum_j f_ij(x_{new})) \tag{2.29}$$

Normally, "one-against-one" method is faster and more memory efficient ($O(N^2)$ and each classifier is much smaller). This method is more convenient to apply a multiclass SVM [62].

2.6 Linear Probabilistic Model

In the literature, There exist other methods that aim to study the classification problem in addition to all the methods discussed in the previous sections. Among these methods, we are interested on ones that build linear probabilistic model in order to predict the classification. In this section, we discuss two very known methods in this case, the Logistic Regression (LR) and the Linear Discriminant Analysis (LDA) [54, 58]. To understand this linear probabilistic classification methods, we define two type of variables:

1. The dependent variables, known also as outcomes, responses variables or the classes noted $Y = \{y_1, \ldots, y_k\}$ where k is the number of classes or outcomes.
2. The Independent variables, known as explanatory or predictor variables.

Linear Regression and Linear Discriminant Analysis are two multivariate statistical methods which are used for analysing categorical outcome variables. Both methodologies try to evaluate the relationships between multiple independents variables and a single categorical dependent variable. Then, Both methods create a linear model that describes the relations between these variables and predict the class label for new observations.

Logistic regression is one of the regression model that is used especially when the dependent variable is categorical. Using the logit function, logistic regression predicts the probability of an outcome in relation to several differents independents variables. In the

other hand, discriminant analysis is a similar classification method that is used to determine which set of variables discriminate between two or more outcome classes and then to classify an observation into these known classes. Using the discriminant functions, the linear combination of the independent variables is used to differentiate between the existing outcomes.

2.6.1 Logistic Regression

The idea of any regression method is to express the dependent variables $(y_i)_{i=1,\dots,k}$ as a linear combination of p independent variables $(\mathbf{x_j})_{j=1,\dots,p}$. Then, it estimates the values of all the coefficients $b_0, \dots, b_n$ in this linear combination defined as follow:

$$y_i = b_0 + b_1 x_{i1} + \cdots + b_n x_{in}$$

The logistic regression creates a predictive model for a categorical dependent variables. It provides a probability scores that predict the values of dependent variables. It is based on calculating the odds of the dependent variable as the ratio of the probability of having this variable divided by the probability of not having it. The logistic regression can be:

1. Binomial, called Binary Logistic Regression, if it deals with two dependent variables
2. Multinomial, called Multinomial Logistic Regression, if it deals with many dependent variables

The Multinomial Logistic Regression model has the following form:

$$ln(\frac{\pi(x)}{1-\pi(x)}) = \mathbf{logit}(\pi(x)) = b_0 + b_1\mathbf{x_1} + \cdots + b_n\mathbf{x_n}$$

where :

$$\frac{\pi(x)}{1-\pi(x)} = \texttt{odds}$$

$$\pi(x) = p(Y = one\ outcome/X_1 = x_1, \dots, X_n = x_n) = \frac{e^{b_0+b_1\mathbf{x_1}+\cdots+b_n\mathbf{x_n}}}{1+e^{b_0+b_1\mathbf{x_1}+\cdots+b_n\mathbf{x_n}}} = \frac{e^c}{1+e^c}$$

The **logit** (π) is the log of the odds ratio or *likelihood* ratio where π can only range from 0 to 1 like any probability. The model of a Logistic Regression looks like any linear regression but it uses a *maximum likelihood* method to get the best fitting model instead

of the ordinary *least squares*. The evaluation of the Logistic Regression is provided by two important concepts:

1. The **odds** : The probability of how much more likely it is that an observation is a member of an outcome rather than a member of the other outcome. Odds value can range from 0 to innity (If the probability (π) is 0.60, the odds are 3 to 2 or 0.60/0.40).
2. The **odds ratio (OR)** : For each coefficient of the regression model, we can calculate a OR as e^b of the predictor x. This odds ratio estimates the change in the odds membership in an outcome for one unit increase in the predictor.

The model fitting in this case is evaluated using the *log likelihood* (LL) ratio test based on $-2LL$ ratio. We are studying the global significance of the model under the following hypothesis:

$$\mathbf{H_0}: \ b_1 = \cdots = b_n = 0$$

$$\mathbf{H_1}: \ \text{one coefficient at least is non null}$$

We apply the following test:

$$-2LL = -2ln(\frac{likelihood \text{ of the fitted model}}{likelihood \text{of the null model}}) \longrightarrow \chi^2_n$$

If the significance is 0.05 level or lower that means the model with predictors is significantly different from the one with the constant only (b_0) and if the significance fails to reach the 0.05 level that means the null hypothesis that knowing the independent variables has no increased effects in predicting the dependent variables.

Indeed, we can test if each one of the predictors represent significantly the Logistic Regression model by using the **Wald** statistic test.

2.6.2 Discriminant Analysis

The Discriminant Analysis (DA) is a multivariate statistical method used to build a descriptive and predictive model of discrimination factors based on observed predictor variables.

Discriminant Analysis is similar to any regression method in the determination of a linear equation that try to predict the outcome of any new observation. It combines the

predictors scores (weights) in some way to seperate between outcomes. The form of the a discriminant function is:

$$\mathbf{D} = \alpha_0 + \alpha_1 \mathbf{x_1} + \cdots + \alpha_n \mathbf{x_n}$$

where $\mathbf{D}$ is discriminante function. α_0 is a constant. $(\alpha_i)_{i=1,\ldots,n}$ are the discriminant coefficients. it is the same as the b's in the Logistic Regression. $(\mathbf{x_i})_{i=1,\ldots,n}$ represent the independent (predictor) variables. And, n is the number of the predictor variables.

Normally, this function try to maximize the distance between outcomes which will minimize the possibility of misclassifying any observation. The number of discriminant function is one less the number of outcomes where there exist one function to seperate between two outcomes. [7] describes how to build the discriminant function for two and more than two outcomes.

There are many different purposes of using Discriminant Analysis like :

- Investigate differences between outcomes based on the predictors in order to discriminate effectively among outcomes.
- Identify important discriminating variables that indicate which predictors contribute most to outcome seperation.
- Classify new observations.

2.7 Quality Measures

Kernel methods have become a very popular pattern analysis algorithms in machine learning. These methods provide high performance in wide variety of supervised and unsupervised learning tasks [131]. A kernel method perform a mapping from the input space into higher dimensional space (feature space). It integrates different type of data sets and detect very general types of relations. There are two reasons to success of these methods, mapping data into a feature space and using a linear algorithm in this new space [132].

The popularity and the high performance of the kernel methods encourage to study the quality of these methods which depend of the goodness of a kernel. The goodness of a kernel function can be studied from the goodness of the kernel matrix K . There are many efficient methods to evaluate the kernel matrix quality especially for classification task such as Kernel Target Alignment (KTA) [25], Feature Space-based kernel Matrix evaluation measure (FSM) [109] and Relative Cut Edge Weight (RCEW) [120].

Consider the training example $\{x_1, \ldots, x_n\} \in \chi$ with the corresponding target vector $\{t_1, \ldots, t_n\} \in \{-1, +1\}$ and the kernel function ϕ defined:

$$\phi : \chi \longrightarrow \mathcal{H}$$

where $\mathcal{H}$ is the feature space which usually reproducing the kernel Hilbert space (RKHS).

Because $\mathcal{H}$ is a high dimensional space, the kernel methods use the kernel trick to operate on the kernel matrix induced from the training example:

$$K = \{k_{ij} =< \phi(x_i), \phi(x_j) >\}_{i,j=1,\ldots,n}$$

2.7.1 Kernel Target Alignment (KTA)

KTA is one of the most commonly used kernel goodness measure because of its simplicity and efficiency. it is used to evaluate the degree of kernel matrix K alignment to its target. This degree is defined as the normalized Frobenius inner product among the kernel matrix and the target vector covariance matrix. This alignment interpreted as cosine distance between these two bi-dimensional vectors.

Let K and K^* be two kernel matrices and $M_{target} = t.t^T$ be the target matrix. The Frobenius inner product of K and K^* is

$$< K, K^* >_F = \sum_{i=1}^{n} \sum_{j=1}^{n} k_{ij} k^*_{ij}$$

For a given kernel K and a target matrix M_{target}, KTA equal:

$$KTA(K, M_{target}) = \frac{< K, t.t^T >_F}{\sqrt{< K, K >_F < t.t^T, t.t^T >_F}}$$

Where $0 \leq KTA \leq 1$

If $KTA(K, t) = 1$, K and M_{target} are linear.

Rule: higher KTA means higher information quality of the kernel K.

KTA is a sufficient condition to be a good kernel matrix but not a necessary condition [109]. So, having a kernel matrix with a very good performance does not mean that KTA is high.

2.7.2 Feature Space-based kernel Matrix evaluation measure (FSM)

The **FSM** goodness measure is created in order to improve and overcome the limitations of the KTA index. In this case, two factors are considered, the within-class variance between class centers and the distance between these class centers using the data distribution in the feature space (see figure 2.15 in page 36). FSM is defined as the proportion of the total within-class standard deviation between the class centers to the distance between the class centers:

$$FSM(k,t) = \frac{std}{||\phi_- - \phi_+||}$$

where *std* is the total within-class standard deviation between the class centers, class positive (ϕ_+) and the class negative (ϕ_-) denoted as the mean of the sample in the feature space:

$$\phi_+ = \sum_{i=1}^{n_+} \frac{\phi(x_i)}{n_+}$$

and

$$\phi_- = \sum_{i=n_++1}^{n_-} \frac{\phi(x_i)}{n_-}$$

Then,

$$std = \sqrt{\frac{\sum_{i=1}^{n_+} < \phi(x_i) - \phi_+, e >^2}{n_+ - 1}} + \sqrt{\frac{\sum_{i=n_++1}^{n_-} < \phi(x_i) - \phi_-, e >^2}{n_- - 1}}$$

where e denote the unit vector of the between two class centers:

$$e = \frac{\phi_- - \phi_+}{||\phi_- - \phi_+||}$$

$FSM(k,t) \geq 0$

Rule: smaller FSM means better quality of kernel matrix.

2.7.3 Relative Cut Edge Weight (RCEW)

This index of goodness, known as the separability index [168], is based on the non parametric statistic the **Cut Edge Weight** (CEW). To calculate the RCEW, we:

1. build a neighborhood graph such as the relative neighborhood graph RNG

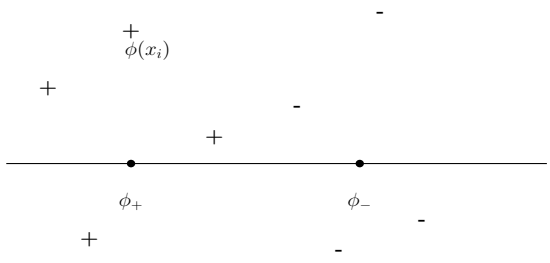

FIGURE 2.15: Data in the direction between-class centers

2. cut all edges between examples of different classes
3. calculate the sum of the cut edges weights J and the sum of the uncut edges weights I

The principle of this procedure is that the classes are well separable (which means the classification method is efficient), if the global weight of the cut edges is very small.

$$J = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} Z_{ij} E_{ij}$$

$$I = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} (1 - Z_{ij}) E_{ij}$$

where W_{ij} = similarity between x_i and x_j and

$$Z_{ij} = \begin{cases} 1 & \text{if } x_i \text{ and } x_j \text{ have the same label} \\ 0 & \text{if not} \end{cases}$$

and

$$E_{ij} = \begin{cases} 1 & \text{if } x_i \text{ and } x_j \text{ are neighbors} \\ 0 & \text{otherwise} \end{cases}$$

Finally, the relative cut edge weight is defined as the proportion of the similarity between neighbors of different classes to the overall similarity of all neighbors :

$$RCEW = \frac{J}{I + J}$$

Rule: smaller RCEW means better quality of the kernel matrix.

We are interested on studying these indices because of its importante role, verify the performance of kernels and algorithms based on neighbrhood graphs. In chapter 3, one of these experiments is described where we use the RCEW index (see subsection 3.4.6).

2.8 Statistical Tests

Research in Machine Learning requires the use of significance tests (see Appendix A.3). These tests are usually used to estimates the performance of an algorithm after applying sampling methods like cross validation and Bootstrap sampling that randomly partition the studied database. Mostly all the researchers in this field were using the `Student` t test to evaluate the performance of their works. However, this test is not efficient in this case because the samples obtained by sampling methods and used to evaluate the proposed algorithm, are not independents (see section 3.4.5 in chapter 3). Therefore, we propose the use of one of these two following tests:

- `Wilcoxon` Signed Ranks test: a non parametric test;
- 5x2cv Paired test: an amelioration of the normal `Student` t test;

2.8.1 Wilcoxon Signed Ranks test

The **Wilcoxon signed ranks test** [156] is a non parametric test used to compare two related data sets. It is used to test the median difference between the two data sets:

$$H_0 : Median_{D_1} = 0$$
$$H_1 : Median_{D_1} \neq 0$$

The wilcoxon signed ranks test statistic W is computed by following the following steps:

1. Compute the difference score D_i,for each individual in the data, between the two related data
2. Find the absolute value of D_i, $|D_i|$
3. Assign ranks R_i to each $|D_i|$
4. Reassign to each R_i the original positive or negative sign of D_i

5. Compute the statistic W as the sum of the positive ranks:

$$W = \sum_{i=1}^{n} R_i^+$$

Then, using the table of exact critical values see appendix A.4, the decision is taking to accept or reject the null hypothesis H_0. For a larger data set, the statistic Z normally distributed is used:

$$Z = \frac{W - \frac{1}{4}n(n+1)}{\sqrt{\frac{1}{24}n(n+1)(2n+1)}}$$

With $\alpha = 0.05$, the null hypothesis H_0 is rejected if Z is smaller than -1.96.

To illustrate this statistical test, we consider two data sets $Data_1$ and $Data_2$ formed of 10 types of flowers. To test if their exist a statistical difference of prices between two stores, we used Wilcoxon test as detailed in table 2.3 page 38.

$Data_1$	$Data_2$	D_i	$\|D_i\|$	$Ranks$	R_i
5	7	-2	2	6	-7
15	10	5	5	10	10
11	11	0	0	1	1.5
9	8	1	1	3	4
8	10	-2	2	7	-7
13	10	3	3	9	-9
12	12	0	0	2	1.5
10	11	-1	1	4	-4
15	17	-2	2	8	-7
11	10	1	1	5	4

TABLE 2.3: Setting up the wilcoxon signed ranks test

$$W = \sum_{i=1}^{10} R_i^+ = 10 + 4 + 4 + 1.5 = 19.5$$

Using the table of the critical values, $19.5 > 8$. So, we accept H_0.

2.8.2 5x2cv Paired test

5x2cv paired test, proposed by Diettreich [32], is a powerful statistical test that have an acceptable type I error (see table 2.4).

Decision / Reality	verified hypothesis	rejected hypothesis
Accepted	right acceptance	Error type II
Rejected	Error type I	right rejection

TABLE 2.4: Test error

This test uses five replications of 2-fold cross-validation. We consider two learning algorithms A and B and we measure their respective accuracies a_ij and b_ij where $i = 1, 2$ and $j = 1, \ldots, 5$. Each learning algorithm is trained on a set and tested on the other set. Let x_j denote the mean difference for a single replication of the 2-fold cross-validation.

$$x\_j = \frac{x_1j + x_2j}{2}$$

The variance is :

$$\hat{\sigma_j^2} = (x_1j - x.j)^2 + (x_2j - x.j)^2$$

The 5x2cv paired test uses the following statistic:

$$t = \frac{x_{11}}{\sqrt{\frac{1}{5}\sum_{j=1}^{5}\hat{\sigma_j^2}}}$$

This statistic follow the student t distribution with five degrees of freedom.

2.9 Conclusion

`Given two or more classes of objects, we try to assign a new object to one of these classes`:

Solving this problem, known as the classification problem, represents the goal of this thesis. Therefore, in this chapter, we describe different approaches to learn from data on solving this problem. We discuss some of the most known and successful methods on Machine Learning such as the Ensemble Method (EM), the Learning Trees, K Nearst Neighbors and Support Vector Machine. These algorithms are known because its provide efficient resolution of the classification problem.

We are interested on studying Ensemble Methods (EM). It is well known that EM can be used for improving prediction performance because a classifier based on EM is often much more accurate than the individual classifiers. In this chapter, we provide a brief survey of methods that apply the Ensembles Methods process such as Adaboost, Random Forests and KNN.

Furthermore, studying the topology of the data can be very helpful to improve the performance of a learning method. This topology can be expressed by using neighborhood graphs (like Gabriel Graph (GG) or Relative Neighborhood Graph (RNG) [147]) which make explicit the neighborhood relation between data observations in the data space. In paper [3], the author explains that many different and useful topological information that we would like to extract from the data may be extracted from a graph. Therefore, the Graph learning can be a new way to study the classification problem because it can provide the decision support.

In the next chapter, we present our method, Random Neighborhood Classification (RNC), that is a combination between the neighborhood graphs and the EM procedure. By combining EM's and neighborhood graphs, we try to construct a powerful classifier that can be efficient and reliable comparing to Random forest and KSVM. Many of the algorithms described in this chapter are used in chapter 3 to evaluate the efficiency of RNC.

Chapter 3

Neighborhood Random Classification

In the last two decades, the Machine Learning algorithms have known great success and progression. Many improvements and different concepts are used to construct efficient algorithm with strong generalisation ability. The Ensemble Learning, one of the very known algorithm in Machine Learning, has proved their efficiency. It tries to achieve strong generalisation by combining multiple classifiers. Indeed, one of the new way to construct a good classifier is that based on the topology of the data [3]. As there is no many work that use the Ensemble Method (EM) based on topological classifier, we are applying this idea. We think that a topological concept with a great and efficient procedure like EM can impove the performance of EM and can be more efficient than all the EM algorithms used until today like AdaBoost and Random Forests.

The EM algorithm generates a set of classifiers using one or several machine learning algorithms and then aggregates them into a single classifier using an aggregation function like the majority voting. EM uses all the efficient information from each classifier and aggregates them to find the ultimate classification result. Many works [17, 164] have shown that a set of classifiers produces a better prediction than the best among them. The efficiency of EMs lies in the fact that aggregating different and independent classifiers reduces the bias and the variance of the aggregate classifier [11, 17, 35, 83], which are two key concepts for effective classifiers. To apply EM, there exist two ways to partition the observation space into regions either randomly as proposed [45, 60, 61, 81, 82] or with according to general rules established by an algorithm such as decision trees as does [13]. Each region represents a local classifier, the prediction rule of the individual local classifier is generally based on the individuals vote of the training sample contained in the region. It is obvious that the way these regions are constructed must

affect each individual classifier and consequently may affect in some way the aggregate classifier. Therefore, we try to use a new way to construct these local classifiers based on topological concept to improve the aggregate classifier.

Recently, the Topological Learning, a very large field of machine learning where we care about the topology of the data set more than about the metric issues, has known a growing interest. Basically, the topology can be expressed via the neighborhood structure [3] which could be induced by the metric, defined using the representation space of observations, or could be given by the user, without any reference to a specific representation space. For instance, Gabriel graphs (Preparata, F. P. and Shamos, M.I (1985)) or relative neighborhood graphs (RNG) (Toussaint, G.T(1980)) or others (see section 3.2.3), provide a neighborhood structure that express, in some way, the topological structure of the data set in multidimensional space.

In this work, we build a classifier based on the topological structure of the data called **Random Topological Classifier** (**RTC**). Then, we apply the EM procedure using this classifier [167]. This is the base of our algorithm called **Random Neighborhood Classification**.

In the literature, the idea of using a neighborhood structure as an EM exists. The k-Nearest Neighbors (kNN) principle, an Instance based (IB) algorithm, is used as an EM algorithm. kNN is a very popular method because of its straightforwardness (see section 2.3 in chapter 2). However, to implement this method, it is simply necessary to define a dissimilarity measure on the set of observations and fix the value of k. And, handling the parameter k might be difficult for some users. Therfore, to simplify this problem, we use approaches based on neighborhood graphs as alternatives. For example, Relative Neighborhood Graphs (RNG), Gabriel Graphs (GG), Minimum Spaning Tree (MST) or Delaunay Triangulation (DT) are a "good" candidates. Like kNN, for an unlabeled observation, the classifier, based on neighborhood graphs, assigns a label according to the labels in the neighborhood. And then, for example, we can simply use the majority voting in the neighborhood of the unlabeled observation to find it class. While many studies using kNN in the context of EM have been carried out, we did not find any study that assesses the interestingness of such neighborhood graphs in EM approaches.

In this chapter, we propose an EM approach based on neighborhood graphs and we provide comparisons between these graphs and then with many different approaches very known in the literature like kSVM, Decision Tree (Random Forests), kNN etc. We have performed our experiments on an R platform. The chapter is organized as follows. In section 3.1, we present and explain the different parts of our approach, the **Random Neighborhood Classification**. Besides the classic kNN neighborhood, The

section 3.2 describes the topological concept and then defines four different neighborhood Graphs. Section 3.3 presents many known aggregation functions used as part of the EM procedure. Section 3.4 is devoted to evaluate our method and compare it to other methods. Section 3.5 gives the main conclusions of this study.

3.1 The General Framework

We propose, in this chapter, an Ensemble Method based on topological classifiers which use the proximity information between individuals (see section 3.2). This framework is called **Random Neighborhood Classification**(**RTC**). It combines between two approaches the topological concept and the Ensemble Method. Our goal is to prove that this combination can be efficient.

3.1.1 Random Neighborhood Classification

To start, we create a new classifier based on the neighborhood graph, called "Random Topological Classifier" (**RTC**). Then, we use the Ensemble Method procedure to evaluate the effeciency of this new classifier. Based on the dataset Ω, we use the learning set E_l to learn from the data and then, the test set E_t to compute the error rate (see section 3.4). We apply the principle of EMs by generating M classifiers using E_l and then aggregating them into one (see figure 3.1). To do so, m randomized iterations are performed. At iteration i, **RTC**:

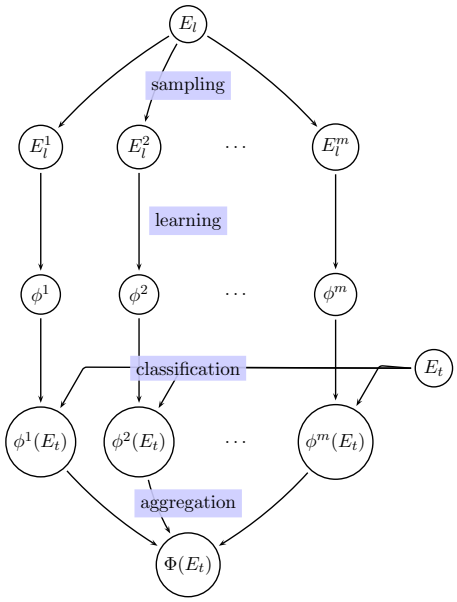

FIGURE 3.1: The EMs based on neighborhood graph

1. generates a new learning set E_l^i with a given size;
2. generates a new classifier $\phi^i = (\mathcal{P}^i, \mathcal{V}^i, C^i)$ (see subsection 3.2.1);
3. uses the generated classifier to determine membership class of unclassified individual in E_t, the test sample.

Following these steps, **RTC** aggregates the m predicted values related to an unclassified individual of E_t to determine its final membership class. The two key points of this procedure are the sampling procedure for generating the m classifiers and the procedure for combining the m predictions. Below, we provide some details about the two key points:

3.1.1.1 Sampling procedures

From the learning data set E_l which is an $n \times p$ table of values, we carry out m random samples. The sampling can be achieved in different ways:

- Sampling on rows with or without replacement;
- Sampling on columns;
- Building new columns by a linear combination of some existing ones (oblique projection);
- Generating new individuals by a linear combination of columns;
- Adding randomly $x\%$ of rows and/or columns.

Each sample produced leads to a specific classifier. The sampling procedure that we have used in our study is described in section 3.4.

3.1.1.2 Aggregting Functions

Generally, the aggregation function is based on the majority rule vote. However, many other possibilities could be used [134]. Among these, we may cite:

- Vote of classifiers, which aggregate the responses of each classifier and normalize them. The majority rule vote is a particular case of this one.
- Average vector where the score for each class is the mean of the answers for all the classifiers.

- Weighted version (majority or mean).
- Maximum Likelihood calculated as the product of the answers for all the classifiers, for each class. The winner class is the one that has the highest value.
- Naive Bayes and Decision Templates (described in section 3.3).
- Linear regression where we assume that the probability of a class is the linear combination of the probabilities of class for each classifier.

3.1.2 Example: Random Forests

Random Forests are a particular case of EM (see [13] and [97] for an implementation). To generate the m classifiers, it performs m bootstrap sample of n individual from the training set. Each bootstrap forms the learning set E_l^i for a tree. Note that another form of randomization is introduced when creating the tree, because, at each node, the variables (columns), which can be selected to cut the node, are chosen in a random subset of the available variables. The minimal entropy tree (the purest) is constructed using bootstrap sample. This tree represents the classifier ϕ^i. To determine the class membership of an individual ω, each classifier ϕ^i take the class of the node in which it falls. Moreover, each classifier gives a binaries vector: $\phi^i(\omega) = (0, \dots 0, 1, 0, \dots, 0)^{\mathrm{t}}$. Then, the aggregated result $\Phi(\omega)$ is obtained using a majority vote :

$$\begin{aligned} \Phi(\omega) &= \sum_{i=1}^{m} \frac{1}{m} \phi^i(\omega) \\ &= (\Phi(\omega)_1, \dots, \Phi(\omega)_i, \dots, \Phi(\omega)_m) \end{aligned}$$

The final result is such that :

$$\Phi(\omega)_f = \max\left(\Phi(\omega)_1, \dots, \Phi(\omega)_i, \dots, \Phi(\omega)_m\right)$$

3.2 Neighborhood Graphs

One major point in the generation of Random Forests is the way to partition the representation space (see subsection 3.2.1). In fact, we can partition the space according to any shape and apply the same principle as random forests. The only problem is to be able to locate the region of any individual in order to predict its class. In this case, Random forests use cuts by parallel or oblique axis which lead to a specific forms of region.

We propose another way to build voting regions [3], it is by directly use the geometric zone formed by the class of the learning set. To do this, we construct a neighborhood basis $\mathcal{P}$ of individuals from the learning sample and a neighborhood function $\mathcal{V}$ which associate new individuals to an element of the neighborhood basis. Using these neighborhood structure, we build the Topological classifiers.

3.2.1 Neighborhood structure

The stucture of a classifier is very important to build it. Therefore, we propose three composants, defined below, to explain the neighborhood structure of a topological classifier.

1. ***neighborhood basis*** $\mathcal{P}$ ***:*** The set of all subsets of the learning set E_l which form a topology cover of E_l. This basis consists of all possible neighbors to which each individual will be connected. In some cases, like Random Forests or our method the RNC, the neighborhood basis is pre-computed on the learning sample. In others, like methods using the rule of k-nearest neighbors, the neighborhood basis is implicit, induced by the neighborhood function

2. ***the neighborhood function*** $\mathcal{V}$ ***:*** This function defines the way in which an individual is linked to an element of the neighborhood basis: $\begin{array}{rcl} \mathcal{V} : \mathbb{R} & \longrightarrow & \mathcal{P} \\ X & \longrightarrow & v = \mathcal{V}(X) \end{array}$
   This function allows to associate any point $X(\omega)$ to a subset of E_l which contains its neighbors. Notes that these neighbors are the only ones used to determine the class of $X(\omega)$.

3. ***the decision rule*** $\mathbf{C}$ ***:*** This rule estimates the distribution of classes

$$\begin{array}{rcl} C : \mathbb{R} \times \mathcal{P} & \longrightarrow & \mathcal{S}_l \\ X, v & \longrightarrow & C_v(X) = (p_1, p_2, \ldots, p_k) \end{array}$$

   where $\mathcal{S}_l = \left\{ (p_1, \ldots, p_k) \in [0,1]^k / \sum_{l=1}^{k} p_l = 1 \right\}$

Hence, we can define a neighborhood classifier ϕ as based on a combination of the triplet $(\mathcal{P}, \mathcal{V}, C)$:

$$\phi(X(\omega)) = C_{\mathcal{V}(X(\omega))}(X(\omega))$$

There are many types of neighborhood structure that could be used to build a topological classifier. Among the most well known are:

- The k-nearest neighbors described in chapter 2 section 2.3.
  - The basis $\mathcal{P}$ is formed by k-elements subsets of E_l,
  - The neighborhood function associates to each individual, the k-nearest neighbors from E_l.
- The ε-neighbors like PARZENs window neighbors, which are defined by the subset of E_l that are in the ball of radius ε, centered on the individual, i.e. a point in Euclidean space.
  - The basis $\mathcal{P}$ is formed by subsets of E_l contained in a ball of radius ε and of the empty set.
  - The neighborhood function associates to each individual the $\varepsilon-$neighbors from E_l.
- The neighborhood regions defined by a decision tree. Each leaf of the decision tree defines a subregion of the space. An individual that falls in a specific leaf has, as neighbors, those of the learning sample located in the same leaf (see figure 3.2).
  - Each point of the learning set is assigned to one of the terminals nodes of the tree, forming the neighborhood basis $\mathcal{P}$. The figure 3.2 proposes a classification tree that generates a partition of $\mathbb{R}^2$ into 5 regions. This tree creates partitions of the training set Ω' which constitute the neighborhood basis.
  - When a new individual ω_{new} occurs, it's easy to locate its neighbors passing by the tree from root to the terminal node. That also induces a partition of the representation space as shown in figure 3.2 where the different regions are delimited by straight lines perpendicular to the axes.
- The neighbors in the sense of a specific property (described in subsection 3.2.3). This is the type of neighbors that we use in our method. For example, the GABRIEL Graphs (GG) neighbors are given by the subset of individuals of the learning sample that fulfill a certain condition. Likewise, we can define the relative neighbors (RN), the minimum spanning tree's (MST) neighbors or the DELAUNAY's polyhedron neighbors and so forth [121].

3.2.2 Local individual classifiers

Once $\mathcal{V}(X(\omega))$, the neighbor of $X(\omega)$, is known according to a topological structure induced in the representation space, it's possible to generate a vector of the simplex $\mathcal{S}_l$. Here are some simple ways:

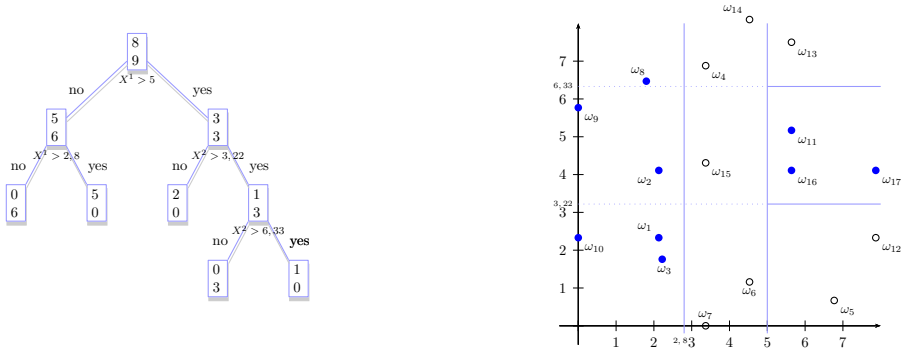

FIGURE 3.2: Partition of the representation space with a tree and the induced neighborhood structure. Each node gives the distribution of the most efficient predicted classes

- *The simple Vote or Frequency* which gives the proportion of each class in $\mathcal{V}(X(\omega))$

$$p(Y = y_k/X) \quad = \quad \frac{\#\left(\{X_i \in \mathcal{V}(X(\omega)) \text{ s.t. } y_i = y_k\}\right)}{\#\left(\mathcal{V}(X(\omega))\right)}$$

- *The weighted Vote :* which allows to take in consideration an information such as the distance between points :

$$p(Y = y_k/X) \quad = \quad \frac{\sum_{X_i \in \mathcal{V}(X(\omega))/y_i = y_k} \frac{1}{1+d(X,X_i)}}{\sum_{X_i \in \mathcal{V}(X(\omega))} \frac{1}{1+d(X,X_i)}}$$

We can binarize these functions by putting 1 for the majority component and 0 for the others: $C(X(\omega)) = (0, \ldots, 0, 1, 0, \ldots, 0)$. Finally, we can choose the class by a simple majority vote or a weighted majority voting.

3.2.3 Neighborhood Graph

Neighborhood graphs, known also as proximity graphs, serve as tools for computational geometry in the field of machine learning. The most simple and natural graph is the k-Nearest Neighbor graph (kNN) where each point in the training set is joined to its nearest neighbor by an edge [41]. Another well known neighborhood graphs are the Minimum Spanning Tree (MST) [162] and the Relative Neighborhood graph (RNG), created in 1980 by G.Toussaint as a tool for extracting the shape of a planar pattern [142, 147, 150]. Besides those graphs mentionned above, two other graphs, Gabriel graph (GG) and Delaunay Triangulation (DT), are well known. All these graphs are retated by the inclusion relationship:

$$1NN \subseteq MST \subseteq RNG \subseteq GG \subseteq DT \tag{3.1}$$

In the literature, there exist other neighborhood graphs that we have not mentioned here. Among these graphs, we cite the α-graphs [39] introduced by Edelsbrunner, Kirkpatrick and Seidel in 1983, the β-skeleton [79] introduced by Kirkpatrick and Radke in 1985 and the sphere of influence graph [148] introduced by Toussaint in 1988.

Many works have done using the neighborhood graphs in the supersived learning field [44, 108, 144–146, 149, 169]. There is a vast literature on neighborhood graphs and its application in pattern recognition (see the work of Jaromczyk and Toussaint [67, 143]). Moreover, many works are devoted to extend these graphs to higher dimensions [3, 37, 96, 139, 153].

The desire to resolve problems in pattern recognition like classification and clustering encourage the work on neighborhood graphs. In the case where we just have a set of points on a plane and we don't know any further information neither about the distribution nor how we can analyse this data. The first thing to do is to find some structure among these points, links or relation that can connect between them. We would like to built an algorithm based on a similarity or a proximity that joins these points in order to obtain a meaningful graph at the end [95, 107]. Analysing this graph will be helpful to classify the data.

Using the Toy example (see chapter 2 section 2.1.1), A neighborhood graph of Ω' described by a vector $\mathbf{x} = x_1, \ldots, x_p$ is a graph whose vertices are the different individuals of Ω'. In such a graph, two individuals a and b are connected by an edge when they are close. This proximity is definied by the neighborhood structure of the graph. The neighborhood structure can be the k-Nearest Neighbors, the Minimum Spaning Tree, the Gabriel graph, the Relative Neighborhood graph or the Delaunay Triangulation. δ denotes the Euclidean distance.

3.2.3.1 The Minimum Spaning Tree (MST)

The Minimum Spanning Tree [117] is a spanning tree that connect all the vertices of a graph and the sum of the euclidean edge lengths (weights) is less than the sum for any other spanning tree (see figure 3.3 3.3 on the following page).

The minimum spanning tree is unique if each edge has a different weight. Many algorithms were created to find the MST like Prim's algorithm and Kruskal's algorithm. The first algorithm was developed by O.Boruvka in 1926 who was working to find an efficient electrical coverage of Moravia in the Czech republic. The complexity of the Minimum Spanning Tree algorithms has increasingly improved [19].

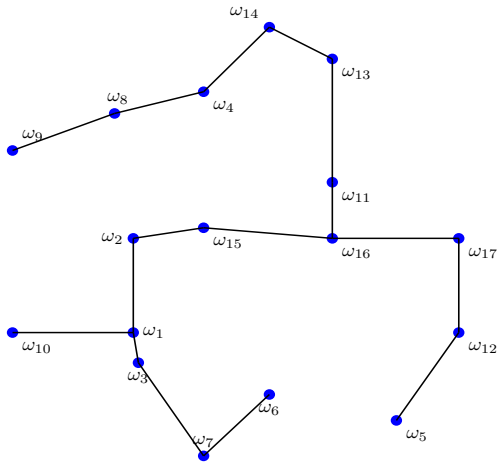

FIGURE 3.3: Minimum Spanning Tree Graph of Ω'

3.2.3.2 The Relative Neighborhood Graph (RNG)

The Relative Neighborhood graph of Ω' is formed of the vertex set Ω' and the set of edges $RNG(\Omega')$ which has two similar definition described as follow :
For any i, j and k in $1, \ldots, m$:

- $RNG(\Omega') = \{(\omega_i, \omega_j) \in \Omega'^2 / \forall \omega_k \in \Omega' - \{\omega_i, \omega_j\}, \delta(\omega_i, \omega_j) \leq max_{\omega_k \in \Omega' - \{\omega_i, \omega_j\}}[\delta(\omega_i, \omega_k), \delta(\omega_j, \omega_k)]\}$
- $RNG(\Omega') = \{(\omega_i, \omega_j) \in \Omega'^2 / \bigwedge_{\omega_i, \omega_j} \bigcap \Omega' = \emptyset\}$
  where $\bigwedge_{\omega_i, \omega_j} = \textbf{lune} = B(\omega_i, \delta(\omega_i, \omega_j)) \bigcap B(\omega_j, \delta(\omega_i, \omega_j))$ is the intersection between two balls centered at ω_i and ω_j (see figure 3.4)

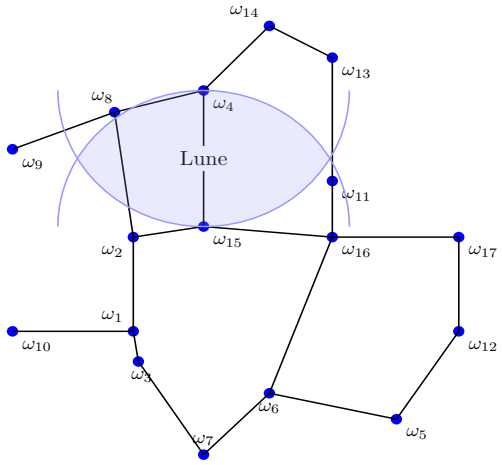

FIGURE 3.4: Relative Neighbors Graph of Ω'

So, if $(\omega_1, \omega_2) \in RNG(\Omega')$ that means that ω_1 and ω_2 are connected by an edge in the RNG graph and we can say that ω_1 and ω_2 are relative neighbors. The RNG is

a superset of the MST and a subset of the DT ($MST \subset RNG \subset GG \subset DT$). This important property can be used for more understanding of RNG graph. The Relative Neighborhood graph is introduced by G.T.Toussaint in 1980 [147] as a perceptually meaningful form of the shape of a finite set of points. In the literature, there are many different algorithms to apply RNG [49, 73]. Indeed, G.T.Toussaint proposed in his work [67, 147] two algorithms for RNG where the first one runs in $O(n^3)$ time and the second in $O(n^2)$ time. His work can be considered as the main bibilography of RNG Graph where many proprities of RNG and different idea to improve the RNG algorithms are presented.

3.2.3.3 The Gabriel Graph (GG)

Gabriel graph was introduced by K.R. Gabriel and R.R. Sokal [50]. The Gabriel graph of Ω' is a graph with vertex set Ω' and edges set $GG(\Omega')$ where all pairs of points (ω_i, ω_j) of $GG(\Omega')$ form the diameter of a disk that doesn't contain other points of Ω'. We define also $GG(\Omega')$ as follow:

$$(\omega_i, \omega_j) \in GG(\Omega') \text{ if and only if } \Gamma_{\omega_i,\omega_j} \bigcap \Omega' = \phi$$

where $\Gamma_{\omega_i,\omega_j} = \textbf{sphere}(ball) = B((\omega_i + \omega_j)/2, \delta(\omega_i, \omega_j)/2)$ (see figure 3.5)

Many algorithms [67, 102, 106] were developed to compute Gabriel Graph. [102] describes several important properties of Gabriel Graph.

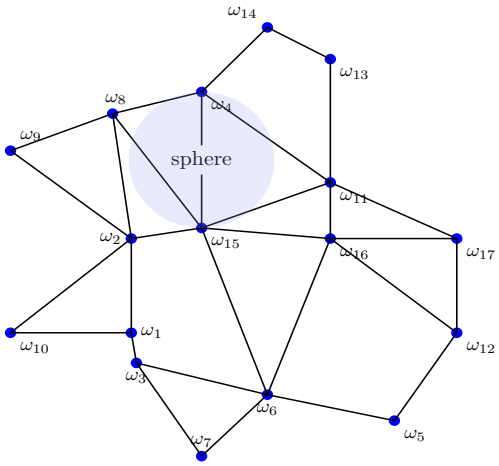

FIGURE 3.5: Gabriel graph of Ω'

3.2.3.4 The Delaunay Triangulation (DT)

The Delaunay Triangulation, proposed by the mathematician B.Delaunay in 1934, is a graph of triangles [38, 99]. The Delaunay triangulation of Ω', $DT(\Omega')$, is a triangulation that contains no points of Ω' in the circumscribed circle formed by the triangles of $DT(\Omega')$ [101, 133](see figure 3.6).

The simplest way to define Delaunay Triangulation is by introducing a very known structure Voronoï diagram [121] , created by G.F.Voronoï [4](see figure3.7).

The Voronoï diagram partitions the plane into disjoint regions, called also tiles or cells, where the tile of each point of Ω' is a set T_i defined by :

$$T_i = T(\omega_i) = Vor_{\Omega'}(\omega_i) = \{t \in \Omega / \forall \omega_{j \neq i} \in \Omega', \delta(t, \omega_i) \leq \delta(t, \omega_j)\}$$

Then, we can easily define the Delaunay Triangulation as follow : two points ω_i and ω_j of Ω' are joined by an edge in the DT graph if and only if the tiles of ω_i and ω_j, T_i and T_j, share a side. Formally,

$$DT(\Omega') = \{(\omega_i, \omega_j) \in \Omega'^2 \text{such that} Vor_{\Omega'}(\omega_i) \bigcap Vor_{\Omega'}(\omega_j) \neq \phi\}$$

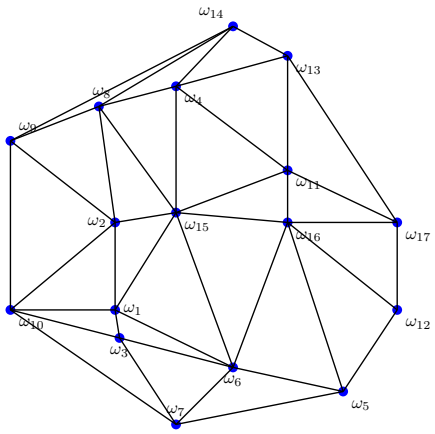

FIGURE 3.6: Delaunay Triangulation Graph of Ω'

3.2.4 Partition by neighborhood graphs

In this work, we are focusing on the Topological graphs described previously. In figure 3.8, we build these neighborhood graphs using the Toy example described in table 2.1. In such graphs, points are linked according to a specific property that defines each graph. This property represents the neighborhood function $\mathcal{V}$.

From these geometric neighborhood graphs, the pure parts are immediately obtained. We simply propose to cut the edges that connect vertices not belonging to the same

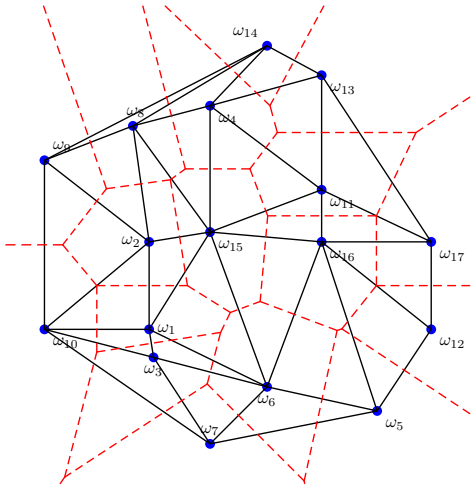

FIGURE 3.7: The Delaunay Triangulation Graph of Ω' defined by the Voronoï structure

class (see figure 3.9). The Connected component of the resulting graphs forms the neighborhood basis $\mathcal{V}$. Comparing to the cut induced by the Random Forests, this approach leads to more larger shape. Recall that in the Random Forests to obtain the pure parts, it is often necessary to develop each tree leading to reduced parts such as singletons. Then, to classify a new point, we can use one of the following three different decision rule **C** based on:

1. Many nearest connected component;
2. The first nearest connected component;
3. The majority voting of all it neighbors;

Figure 3.10 describes these three different decision rules using the Gabriel Graph.

To evaluate our method, we used the majority voting of all neighbors to assign a label class to a new point.

At this stage, the first part of our method is almost finished by having the $\mathcal{V}(\omega_{new})$ function. The second step, explained in the section 3.1, is similar to other Ensemble Method :

- Calculate the estimation, proportion of each class.
- Aggregate the results using the aggregation methods

Below, we describe some of the very known aggregation function.

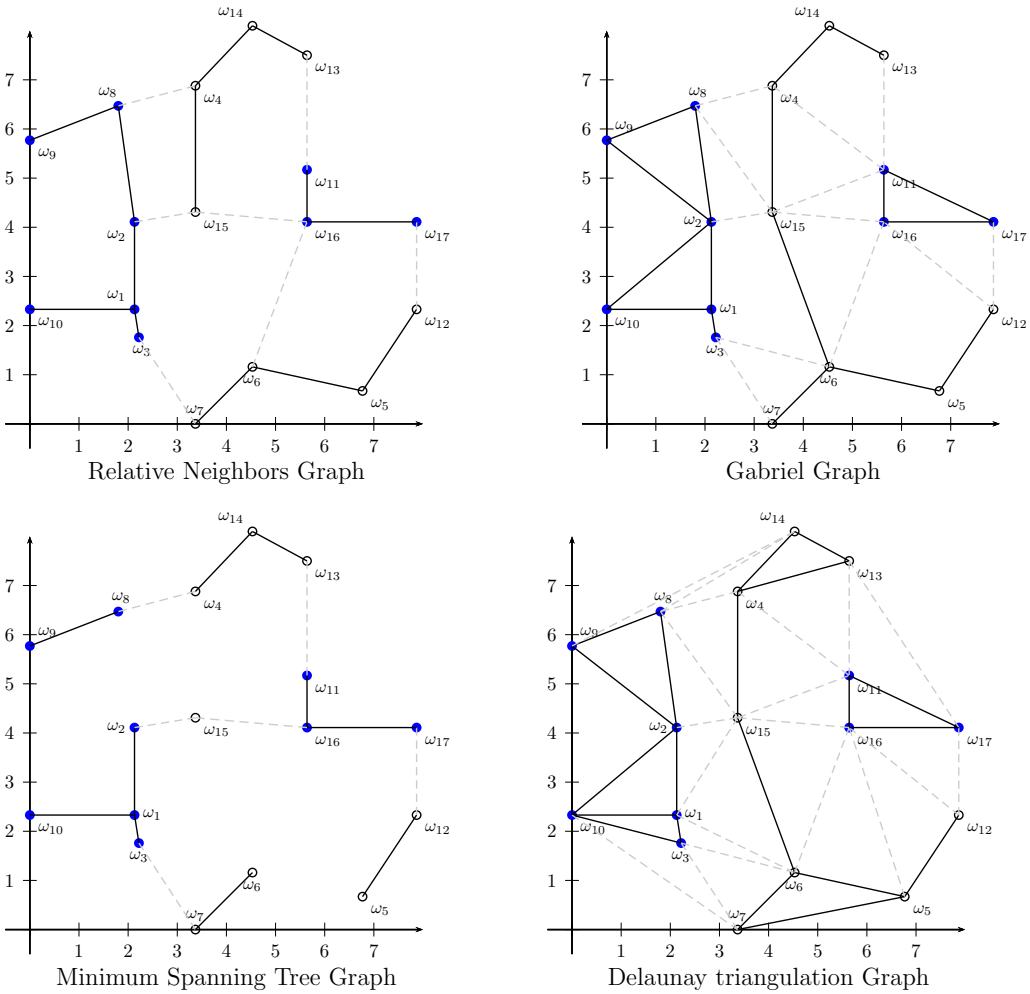

FIGURE 3.8: Four different Neighborhood Graphs: the neighbor structures of the Relative Neighbors graph, the GABRIEL graph, the Minimum Spanning Tree graph and the Delaunay graph, using the dataset Ω' introduced in table 2.1

3.3 Aggregation Methods

Recently, combining classifiers has been viewed as a new direction for the development of highly reliable pattern recognition systems. Results indicate that the combination of several classifiers leads to classifiers that achieve higher accuracy with improved performance [91, 160]. Thus, combining classifiers has become an important research topic.

In the literature, combining classifiers has different names, combination methods [87], combination of multiple classifiers [64, 80, 92, 160], classifier fusion [52, 89], mixture of experts [66] and so on.

In general, combination of classifiers has two basic approaches : classifier selection or classifier fusion [159]. The classifier selection algorithm is based on the assumption that, among all the classifiers, one classifier can predict the most likely correct class label.

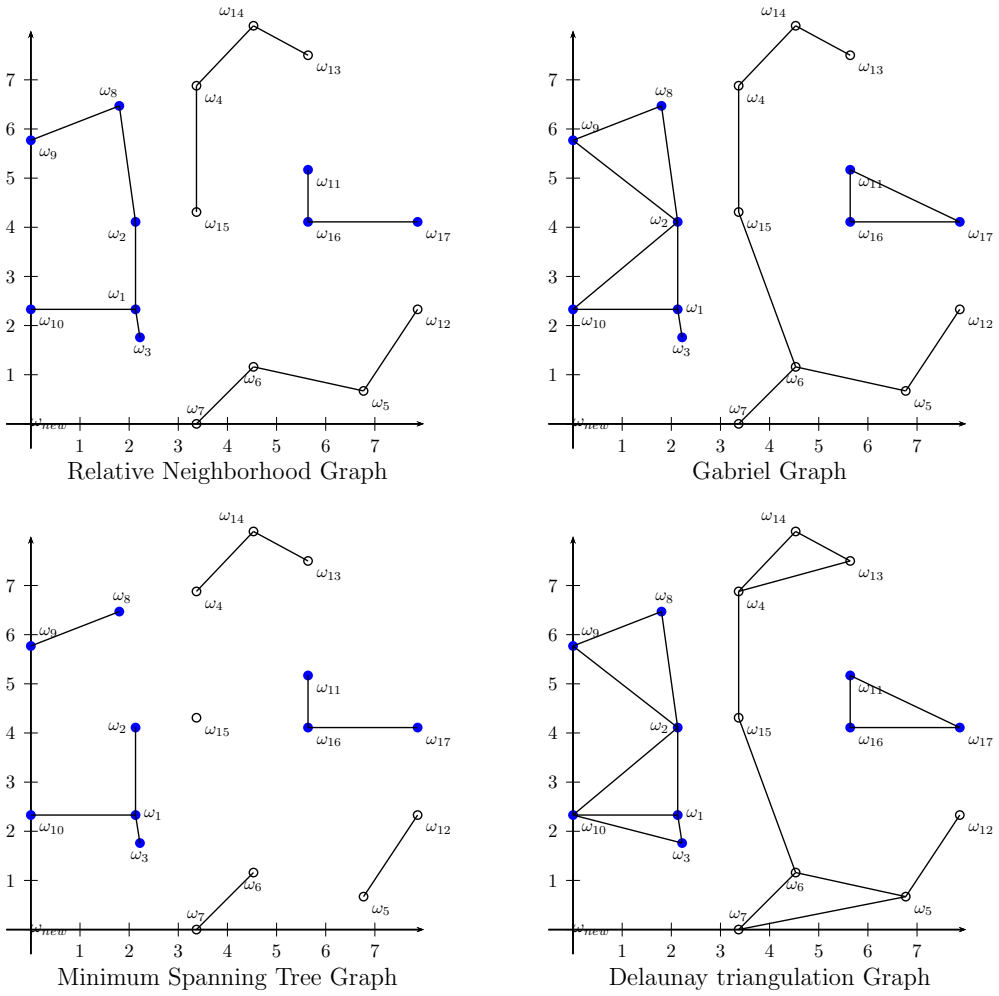

FIGURE 3.9: Partition by neighborhood graphs

The output of this classifier is used in the final decision. If this only classifier does not exist, the classifier fusion is used and the outputs of all the classifiers are combined to achieve the perfect prediction of the class label.

Usually, Classifier's outputs can be compared by scaling them to the $[0, 1]$ interval. For some classifiers these values can be treated as the conditional posterior probabilities for the classes [36]. Furthermore, the interpretation of these probabalities [80, 159] does not work without the independence assumptions. Under such assumptions, classifiers fusion reduces to simple aggregation operators such as the product or average, or the so-called Naive Bayes rule explained in subsection 3.3.3.

Another way to solve the fusion problem is by treating the classifier outputs as the input to a second level classifier. The method that we used in our evaluation is based on a set of matrices called Decision Templates (DecTs). DecTs are a robust classifier fusion scheme that combines classifier outputs by comparing them to a characteristic template for each class. Decision Templates fusion uses all classifier outputs to calculate the final

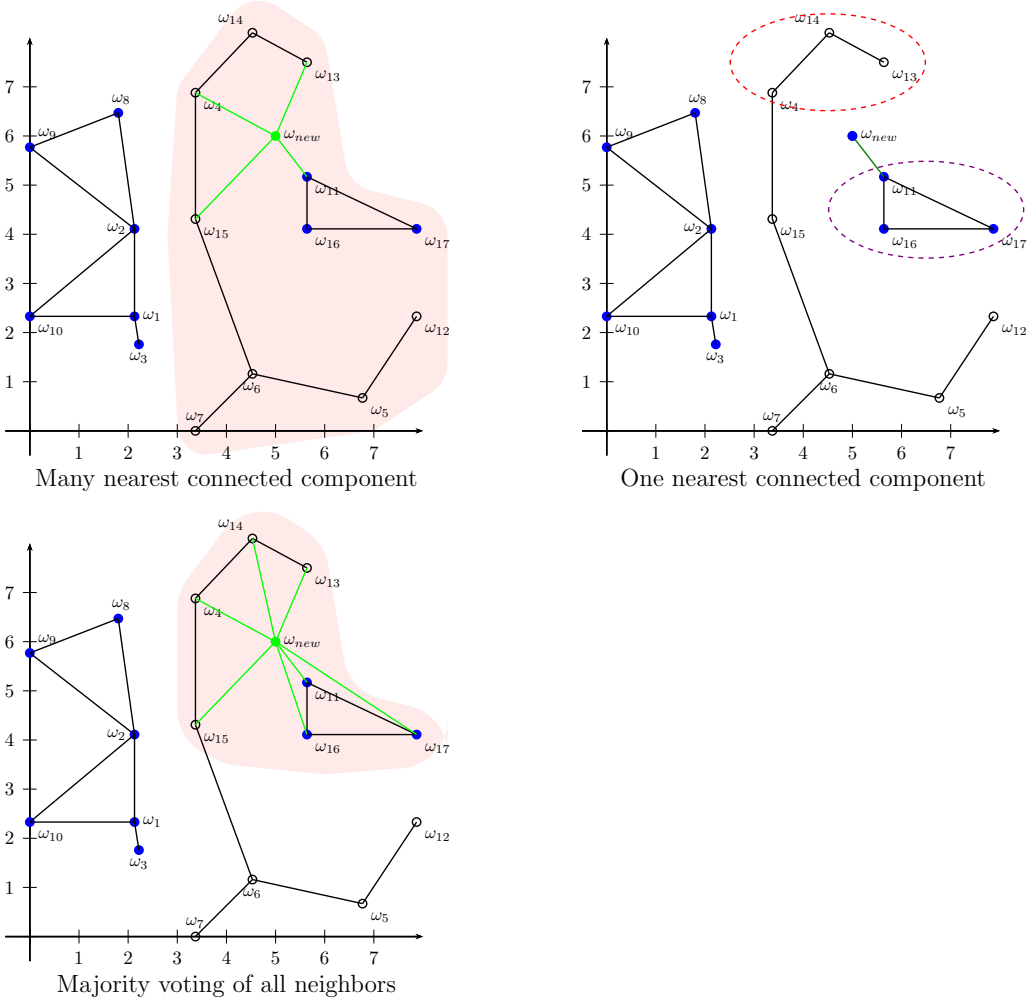

FIGURE 3.10: The three decision rule to affect a new point ω_{new} using Gabriel Graph

support for each class, which is in contrast to most other fusion methods which use only the support for that particular class to make their decision [88].

Following, we describe some aggregation functions, the simple and weighted methods, the Naive Bayes and the Decision Templates.

Let $\omega \in \mathbb{R}^n$ be a feature vector, a new individual. We define a classifier ϕ as the mapping

$$\begin{array}{rcccl} \phi & : & \Re^n & \longrightarrow & [0,1]^k \\ & & \omega & \longmapsto & \phi(\omega) = [d_1(\omega), \ldots, d_k(\omega)]^T \end{array}$$

where $d_i(\omega)$ is the degree of support given by classifier ϕ to the hypothesis that ω comes from class y_i. These degrees of support can be interpreted in differents ways, the most often used is the estimates of the posterior probabilities for classes. We call the output of ϕ a class label:

$$\mu_\phi(\omega) = [\mu_\phi^1(\omega), \ldots, \mu_\phi^k(\omega)]$$

where $\mu_{\phi}^{i}(\omega) \in [0,1]$ is an estimate of the posterior probability $p(y_i/\omega)$.

Let $\phi_1, \ldots, \phi_L$ be a set of L classifiers. The ith classifier outputs k degrees of support $\phi_i(\omega) = [d_{i,1}(\omega), \ldots, d_{i,k}(\omega)]^T$.

The larger support is the more likely the class label. The L classifier outputs for $\omega \in \Re^n$ can be organised in a decision profile ($DP(\omega)$) matrix (see table 3.11)

Definition 3.1. *crisp* is a function defined as following:

$$crisp(a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n) = i \quad \text{if and only if} \quad \forall j \neq i \; a_i \geq a_j$$

where the *crisp* function is different from the Max function:

$$Max(a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n) = a_i \quad \text{if and only if} \quad \forall j \neq i \; a_i \geq a_j$$

$$\mathcal{DP}(\omega) = \begin{pmatrix} d_{1,1}(\omega) & \ldots & d_{1,j}(\omega) & \ldots & d_{1,k}(\omega) \\ \vdots & & \vdots & & \vdots \\ d_{i,1}(\omega) & \ldots & d_{i,j}(\omega) & \ldots & d_{i,k}(\omega) \\ \vdots & & \vdots & & \vdots \\ d_{L,1}(\omega) & \ldots & d_{L,j}(\omega) & \ldots & d_{L,k}(\omega) \end{pmatrix}$$

Support from $\phi_1, \ldots, \phi_L$ for class y_j

Output of classifier $\phi_i(\omega)$

FIGURE 3.11: DP(ω)

We try to construct $\hat{\phi}(\omega) = f(\phi_1(\omega), \ldots, \phi_L(\omega)) = [\mu_1(\omega), \ldots, \mu_L(\omega)]^T$ where f is the aggregation function.

3.3.1 Simple methods

Majority, Maximum, Minimum, Average and product represent the simple aggregation methods because its not require any parameters to be trained.

On one hand, the Majority method combines the classifiers outputs by using the voting principle. The class which receives the largest number of votes is then selected as the majority decision, see figure 3.12.

On the other hand, the Max, Min, Average and product methods have the same procedure. Its defferenciate by the operation (Maximum, Minimum, Summation, Product)

used to get the class assigned by each classifier.

$$\mu_j(\omega) = H(d_{1,j}(\omega), \ldots, d_{L,j}(\omega))_{j=1,\ldots,k}$$

where H is the respective operation: Maximum, Minimum, Average(Sum) or Product. Then, the class y_j with the maximum μ_j is the assigned class, see figure 3.12 in page 58.

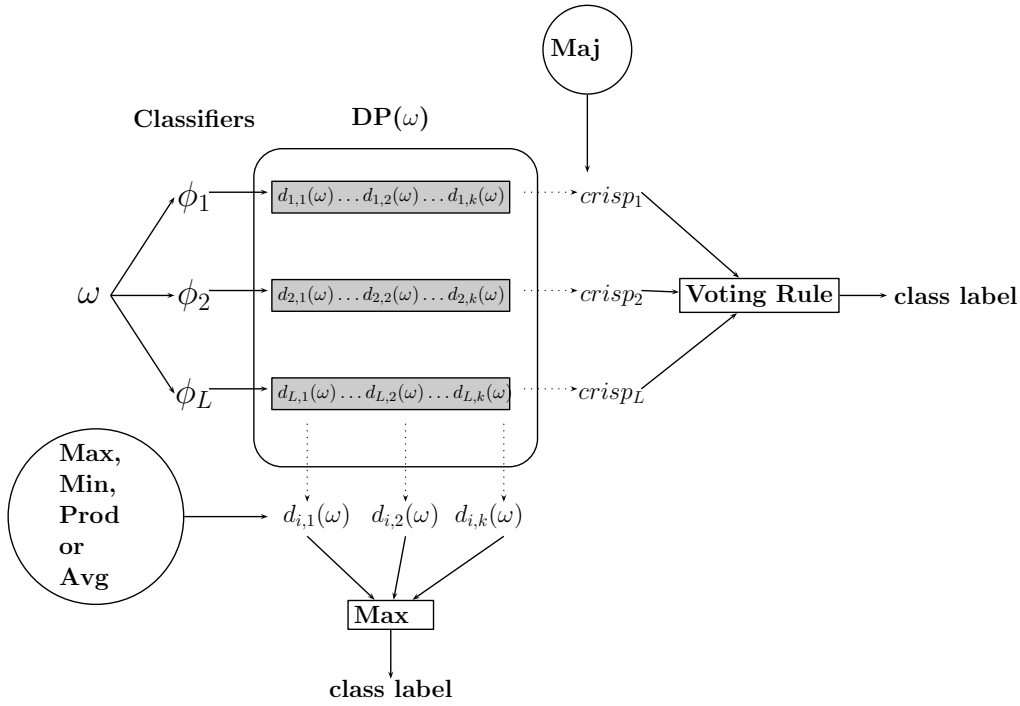

FIGURE 3.12: The simple aggregation methods

3.3.2 Weighted majority or mean

The weight in those methods are supposed to be the ability of each classifier to well classify a particular individual. To estimate this ability, we may use the success of the classifier on the neighbors.

For each individual ω, we use a neighborhood graph of ω in the training set. Then, the ability $a^i(\omega)$ of the classifier ϕ_i on ω is the proportion of well classified individual among those neighbors. Finally, the mean and majority function are weighted by their ability.

ϕ_1	estimated		ϕ_2	estimated		ϕ_3	estimated	
True	s_1	s_2	True	s_1	s_2	True	s_1	s_2
y_1	9	4	y_1	11	2	y_1	8	5
y_2	1	16	y_2	3	14	y_2	11	6

TABLE 3.1: Confusion matrices of respectively ϕ_1, ϕ_2 and ϕ_3

3.3.3 Naive Bayes (NB)

Many works are done in the context of Bayes combination like [77, 89, 135]. Adding "Naive" to the name of this method come from the assumption that the classifiers are mutually independent.

Let $s_1, \ldots, s_L$ be the estimate class labels assigned to $\omega \in \mathbb{R}^n$ by the $\phi_1, \ldots, \phi_L$ classifiers respectively. Using the independance assumption, we calculate the estimate probability of ω in y_j :

$\mu_j(\omega) = \prod_{i=1}^{L} \hat{p}(y_j/\phi_i(\omega) = s_i)$

$$where\ \hat{p}(y_j/\phi_i(\omega) = s_i) \quad = \quad \frac{\#\ objects\ labeled\ s_i\ by\ \phi_i/\ true\ label\ y_j}{\#\ objects\ labeled\ s_i\ by\ \phi_i}$$

ω is attributed to the class with the highest probability.

Naive Bayes: A simple example

To explain the details of NB method, we used a simple example, a data Z of 30 elements with 3 classifiers (ϕ_1, ϕ_2, ϕ_3) and 2 classes (y_1, y_2). The confusion matrices of the three classifiers are as shown in table 3.1

The output of the three classifiers for $\omega \in \Re^n$ is $[s_1, s_2, s_3] = [y_1, y_2, y_2]$. Using the NB method, the probability and results are shown in table 3.2. Then, we conclude that :

$$\mu_1(\omega) > \mu_2(\omega)$$

Probability	$P^*(y_1/s_1 = y_1) = 9/10$ $P^*(y_1/s_2 = y_2) = 2/16$ $P^*(y_1/s_3 = y_2) = 5/11$	$P^*(y_2/s_1 = y_1) = 1/10$ $P^*(y_2/s_2 = y_2) = 14/16$ $P^*(y_2/s_3 = y_2) = 6/11$
Result	$\mu_1(\omega) = 9/10 * 2/16 * 5/11 = 0.0511$	$\mu_2(\omega) = 1/10 * 14/16 * 6/11 = 0.047$

TABLE 3.2: Naive Bayes Example: probability calculations

So, class y_1 will be assigned by Naive bayes method to ω.

3.3.4 Decision Templates (DecT)

This method [86] uses all the values in DP(ω) (Decision Profile) matrix to find the over all support for each class and label the input ω with the class that has the largest support. Two steps describe this method:

1. Calculate the most typical decision profile for each class y_j, called decision template $DecT_j(\Omega)$ for $j = 1, \ldots, k$ and $\Omega = (\omega_r)_{r=1,\ldots,N}$ the training data set. To simplify the notation $DecT_j(\Omega)$ will be denoted by $DecT_j$. The decision template $DecT_j$ of class y_j is the L x k matrix $DecT_j = [dt_j(i,j)]$ whose (i,j)th element is computed as follow :

$$dt_j(i,j)(\Omega) = \frac{\sum_{r=1}^{N} Ind(\omega_r, j) d_{i,j}(\omega_r)}{\sum_{r=1}^{N} Ind(\omega_r, j)}$$

   where $i = 1, \ldots, L$, $j = 1, \ldots, k$ and $Ind(\omega_r, j)$ is an indicator function.

$$Ind(\omega_r, j) = \begin{cases} 1 & \text{if } \omega_r \text{ has label j} \\ 0 & \text{otherwise} \end{cases}$$

   The decision template $DecT_j$ for class y_j is the average of the decision profiles of the elements of the training set Ω labeled in class y_j.

2. Compare the most typical decision profile with the DP(ω). To do so, we calculate the similarity SM between $DP(\omega)$ and each $DecT_j$ based on the Euclidean distance between the DP and $DecT_j$ matrices :

$$\begin{aligned} \mu_j(\omega) &= SM(DP(\omega), DecT_j)_{j=1,\ldots,k} \\ &= 1 - \frac{1}{B \times C} \sum_{i=1}^{L} \sum_{j=1}^{k} [DT_j(i,j) - d_{i,j}(\omega)]^2 \end{aligned}$$

where $DecT_j(i,j)$ is the (i,j)th entry in the decision template $DecT_j$, B is the number of classifiers and C is the number of class. We can drop the scaling coefficient $\frac{1}{B \times C}$. Indeed, we can use other distance like Minkowski or Mahalanobis norm.

If the similarity between the DP matrix and the Decision templates matrix of class j ($DecT_j$) is the highest, then, the class j is considered as the class label. This similarity measure means the degree of matching between DP and $DecT_j$. There exist many different similarity measure [89] that might be appropriate to compare between these two matrices.

Decision Templates: A simple example

Let Z be a training set with two classes and three classifiers. Assuming that we obtained the following Decision Templates $DecT_1$ and $DecT_2$:

$$\mathbf{DecT_1} = \begin{pmatrix} 0.5 & 0.5 \\ 0.6 & 0.4 \\ 0.3 & 0.7 \end{pmatrix} \mathbf{DecT_2} = \begin{pmatrix} 0.1 & 0.9 \\ 0.4 & 0.6 \\ 0.8 & 0.2 \end{pmatrix}$$

Given an element x with the profile $DP(x)$, we can compute the similarity as follow:

$$\mathbf{DP(x)} = \begin{pmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \\ 0.4 & 0.6 \end{pmatrix}$$

$$\begin{aligned} \mu_1(\omega) &= 1 - [d_E(DP(\omega), DecT_1)] \\ &= 1 - [(0.6-0.5)^2 + (0.4-0.5)^2 + (0.2-0.6)^2 + (0.8-0.4)^2 + (0.4-0.3)^2 + (0.6-0.7)^2] \\ &= 1 - 0.36 \\ &= 0.64 \end{aligned}$$

$$\begin{aligned} \mu_2(\omega) &= 1 - [d_E(DP(\omega), DecT_2)] \\ &= 1 - [(0.6-0.1)^2 + (0.4-0.9)^2 + (0.2-0.4)^2 + (0.8-0.6)^2 + (0.4-0.8)^2 + (0.6-0.2)^2] \\ &= 1 - 0.9 \\ &= 0.1 \end{aligned}$$

Since $\mu_1(\omega) > \mu_2(\omega)$, ω is labeled in y_1.

3.4 Evaluations

To assess the performance of our classifier **RTC**, we carried out many experiments on different data sets taken from the *UCI* Irvine repository, the well known and intensely employed by the scientific community, which originate from real world scenarios. For

this, we made some distinctions according to the type of neighborhood used. As our work was motivated by the absence of studies on Ensemble Method based on geometrical graphs, we designed many separate experiments for **RTC** based, respectively on RNGs, GGs, MSTs graphs and the other on kNN where $k = 1, 3, 10$. The comparisons was also extended to random forests (RFs), Support Vector Machines (KSVMs), Adaboost, Discriminant Analysis (DA), Logistic Regression (LR) and C4.5. All these experiments were done using **R** software. The same training data has been used to compare the performances of these methods.

3.4.1 Implementation of Random Topological Classifier (RTC)

The experiments of **RTC** according to the different neighborhood graphs have carried out by using 100 bootstrap samples on each data set. To each bootstrap sample, we built the three neighborhood graphs (RNG, GG and MST). Hence, this allows to build three RTCs for each bootstrap sample that are evaluated on the test sample E_t.

- **Sampling** : variables and individuals of learning sets are together sampled by bootstrap. As mentioned previously, 100 bootstrap samples have been generated :
  - For individuals:
    - if the number of individuals > 1000, we used a sample with the proportion of 66%.
    - if the number of individuals < 1000, we used a bootstrap sample.
  - For variables: the proportion was:
    - 50% of the variable for small dimensions (i.e. the dimension p is less than 20).
    - $\frac{10}{p}$ for the others.
- **Distance between individual** : The three graphs (RNG, GG and MST) have been built according the Mahalanobis distance between individuals.
- **Classification** :
  - The neighborhood of a point $X(\omega) \in E_t$ is computed according to each of the three neighborhood graph :

$$\mathcal{V}(X(\omega)) \quad = \quad \mathcal{V}_{RNG,GG,MST}(X(\omega))$$

– The decision rule is the proportion of each class in the neighborhood :

$$\begin{aligned} \phi(X(\omega)) &= (p_1, \ldots, p_K) \\ \text{such that} \quad p_i &= \frac{\#\left(\{X(\omega_j) \in \mathcal{V}(X(\omega))/Y(X(\omega_j)) = i\}\right)}{\#\left(\mathcal{V}(X(\omega))\right)} \end{aligned}$$

where $\#(\cdot)$ is the cardinal of a set.

- **Aggregation** : The results are aggregated using many different aggregation methods.

3.4.2 Implementation of the other methods

Many algorithms have been used on the same training data sets with the following procedures using the `R` software :

- For kNN, we applied three evaluations for $k = 1,\ 3\ and\ 10$ using the same procedure as the three geometrical graphs described previously. We used the `knn` function of the R `class` library.
- For Random Forests (RF), we used `randomForest` library and we carried out 500 iterations that led to 500 trees.
- For Support Vector Machine, we used the R `kernlab` library to apply the KSVM algorithm with classification type C-svc.
- For Discriminant analysis (DA) and Logistic Regression (LR), we used the `lda` and `glm` functions of the R `MASS` library.
- For AdaBoost and C4.5, we used the `J48` and `adaboostM1` functions of the R `RWeka` library using the control of the Weka learner.

3.4.3 Validation process

First, we used 22 quantitative data sets from *UCI* Irvine repository to apply the implementation of the **RTC** classifier based on RNG, GG and MST using many different aggregation functions. Our goal at the first part of the evaluation is find the best **RTC** classifier. So, we applied all the different combination between the three neighborhood graphs and the following aggregation functions: Naive Bayes (NB), Decision Templates (DecT), Product (Prod), Maximum (Max), Minimum (Min), Median, Mean, Weighted Mean (W_ Mean), Majority (Maj), Weighted Majority (W_Maj) and Summation (Sum).

In tables 3.3 and 3.4, we compared all these combinations. We found that RNG using the Decision Templates aggregation function has the first rank six times. Indeed, we can see that the performance of all these combinations based on neighborhood graph is quite effective. In table 3.5, we computed the mean of error rate and the rank mean for all the different combinations compared in the previous two tables 3.3 and 3.4. The **RTC** based on RNG graph et DecT aggregation function is the best combination. In the following tests, this combination will represent our classifier, noted $\mathbf{RTC}_{RNG}$.

The second part in our evaluation is to compare our method represented by $\mathbf{RTC}_{RNG}$ to other very known method in the literature. Therefore, we ran the protocol described in 3.4.2 over all the methods, Random Forests (RF), K-Nearest Neighbors (KNN) with k=1,3,10, Support Vector Machine (SVM), AdaBoost, Discriminant Analysis (DA), Logistic Regression (LR) and C4.5. To start , we computed the error rate of all the methods. To do se, we applied 10-Cross Validation for each experiment (which mean that each data set is divided into 10 equal parts, 9 parts form the training set and 1 part forms the test set. Each test was repeated 10 times to obtain an estimation of the error rates). Then, we compared between our classifier $\mathbf{RTC}_{RNG}$ and all the other methods. This comparison is evaluated by using two statistical tests, the paired t Student test and the Wilcoxon signed ranks test.

The results are presented in Table 3.6 where, for each data set, we find the average error rate, the rank of each method among the others and the $p$values detected by the Wilcoxon test and the t Student test. Based on this results, we can see that $\mathbf{RTC}_{RNG}$ performs very well in comparison to kNN and also in comparison to the other methods. Indeed, from the 18 data sets, $\mathbf{RTC}_{RNG}$ is ranked first six times, second seven times and third once. So, mostly $\mathbf{RTC}_{RNG}$ is one of the three first very performant methods. In table 3.7, we computed the Mean Error and the Mean Rank of the Error rates and the Ranks found in table 3.6. Our classifier $\mathbf{RTC}_{RNG}$ ranked the first which prouve the importance of our method. Indeed, to see if the difference is significant among all these methods, we applied the simple Friedman test [29], which showed a significant difference among all the methods with a p-value of 2.555×10^{-10}.

3.4.4 Computational analysis

The theoretical complexity for graph computation using n individuals represented by p variables is $O(n^3 + n^2p^2)$. Indeed, for each pair of individuals ω_1, ω_2, it is necessary to test the RNG condition :

$$\forall \omega \in E_l - \{\omega_1, \omega_2\} \quad d(\omega_1, \omega_2) \quad \leq \quad \max\left(d(\omega_1, \omega), d(\omega, \omega_2)\right); \tag{3.2}$$

		Arcene_expurge		Bupa		Ecoli		Glass		Ionosphere		Iris	
		% Err.	Rank	% Err.	Rank	% Err.	Rank	% Err.	Rank	% Err.	Rank	% Err.	Rank
RNG	NB	0.34	19	0.43	21	0.47	24	0.63	28	0.054	10	0.33	25.5
	DecT	0.2	**1.5**	0.3	12	0.13	**1**	0.25	4	0.051	6	0.04	9
	Prod	0.44	29.5	0.56	27.5	0.57	30.5	0.63	28	0.15	26.5	0.35	27.5
	Max	0.44	29.5	0.56	27.5	0.43	21.5	0.56	21.5	0.15	26.5	0.19	23.5
	Min	0.44	29.5	0.56	27.5	0.57	30.5	0.63	28	0.15	26.5	0.35	27.5
	Median	0.44	29.5	0.56	27.5	0.43	21.5	0.56	21.5	0.15	26.5	0.19	23.5
	Mean	0.22	4	0.29	9	0.14	3	0.26	7.5	0.054	10	0.04	9
	W_Mean	0.23	7	0.27	3	0.16	7	0.25	4	0.051	6	0.033	3
	Maj	0.26	12.5	0.28	6	0.19	13.5	0.26	7.5	0.049	2.5	0.04	9
	W_Maj	0.24	10	0.29	9	0.19	13.5	0.26	7.5	0.049	2.5	0.033	3
	Sum	0.22	4	0.33	18	0.36	19	0.34	16.5	0.069	21	0.04	9
GG	NB	0.36	23	0.42	19.5	0.46	23	0.61	25.5	0.077	24	0.37	29
	DecT	0.24	10	0.3	12	0.15	5	0.25	4	0.069	21	0.033	3
	Prod	0.36	23	0.45	23.5	0.52	26.5	0.6	23.5	0.066	17.5	0.18	21.5
	Max	0.36	23	0.45	23.5	0.34	17.5	0.47	19.5	0.069	21	0.13	19.5
	Min	0.36	23	0.45	23.5	0.52	26.5	0.6	23.5	0.069	21	0.18	21.5
	Median	0.36	23	0.45	23.5	0.34	17.5	0.47	19.5	0.069	21	0.13	19.5
	Mean	0.29	16	0.28	6	0.16	7	0.29	12	0.057	14	0.04	9
	W_Mean	0.28	15	0.29	9	0.17	10	0.29	12	0.054	10	0.04	9
	Maj	0.32	18	0.27	3	0.19	13.5	0.31	14.5	0.066	17.5	0.04	9
	W_Maj	0.31	17	0.27	3	0.19	13.5	0.31	14.5	0.063	16	0.033	3
	Sum	0.26	12.5	0.31	14.5	0.39	20	0.39	18	0.16	29	0.033	3
MST	NB	0.35	20	0.42	19.5	0.48	25	0.61	25.5	0.049	2.5	0.33	25.5
	DecT	0.2	1.5	0.3	12	0.14	3	0.24	2	0.054	10	0.047	14
	Prod	0.44	29.5	0.58	31.5	0.58	32.5	0.67	32.5	0.24	31.5	0.67	32.5
	Max	0.44	29.5	0.58	31.5	0.56	28.5	0.65	30.5	0.24	31.5	0.57	30.5
	Min	0.44	29.5	0.58	31.5	0.58	32.5	0.67	32.5	0.24	31.5	0.67	32.5
	Median	0.44	29.5	0.58	31.5	0.56	28.5	0.65	30.5	0.24	31.5	0.57	30.5
	Mean	0.22	4	0.28	6	0.14	3	0.26	7.5	0.057	14	0.047	14
	W_Mean	0.23	7	0.25	1	0.16	7	0.23	1	0.057	14	0.047	14
	Maj	0.27	14	0.32	16.5	0.17	10	0.29	12	0.054	10	0.06	17.5
	W_Maj	0.24	10	0.31	14.5	0.17	10	0.27	10	0.049	2.5	0.06	17.5
	Sum	0.23	7	0.32	16.5	0.32	16	0.34	16.5	0.051	6	0.053	16

		Letter_RvsB		Magic_telescope		Musk1		Parkinsons		Pima		Ringnorm	
		% Err.	Rank	% Err.	Rank	% Err.	Rank	% Err.	Rank	% Err.	Rank	% Err.	Rank
RNG	NB	0.0072	1	0.22	20	0.14	15	0.13	18	0.35	28.5	0.029	8.5
	DecT	0.011	9.5	0.14	**1**	0.085	**1**	0.063	7	0.26	11	0.024	3.5
	Prod	0.19	27.5	0.32	27.5	0.39	27.5	0.38	27.5	0.35	28.5	0.13	27.5
	Max	0.19	27.5	0.32	27.5	0.39	27.5	0.38	27.5	0.35	28.5	0.13	27.5
	Min	0.19	27.5	0.32	27.5	0.39	27.5	0.38	27.5	0.35	28.5	0.13	27.5
	Median	0.19	27.5	0.32	27.5	0.39	27.5	0.38	27.5	0.35	28.5	0.13	27.5
	Mean	0.011	9.5	0.17	7	0.091	7	0.079	10	0.25	4.5	0.04	15
	W_Mean	0.0099	5.5	0.17	7	0.089	4.5	0.063	7	0.26	11	0.042	17
	Maj	0.013	13	0.21	18	0.11	10	0.068	9	0.26	11	0.027	6.5
	W_Maj	0.0099	5.5	0.2	14.5	0.11	10	0.063	7	0.26	11	0.024	3.5
	Sum	0.016	14	0.15	3	0.1	8	0.13	18	0.27	15.5	0.031	11
GG	NB	0.037	18.5	0.2	14.5	0.25	24	0.24	25	0.31	19	0.023	2
	DecT	0.033	15	0.15	3	0.12	12	0.13	18	0.24	1	0.019	1
	Prod	0.039	20.5	0.29	23.5	0.19	21	0.12	13.5	0.33	21.5	0.033	12
	Max	0.04	23	0.29	23.5	0.18	17.5	0.12	13.5	0.33	21.5	0.048	21
	Min	0.04	23	0.29	23.5	0.18	17.5	0.12	13.5	0.33	21.5	0.048	21
	Median	0.04	23	0.29	23.5	0.18	17.5	0.12	13.5	0.33	21.5	0.048	21
	Mean	0.034	16	0.19	11.5	0.19	21	0.18	22.5	0.25	4.5	0.029	8.5
	W_Mean	0.035	17	0.19	11.5	0.19	21	0.17	21	0.25	4.5	0.03	10
	Maj	0.039	20.5	0.2	14.5	0.26	25	0.19	24	0.25	4.5	0.025	5
	W_Maj	0.037	18.5	0.2	14.5	0.24	23	0.18	22.5	0.25	4.5	0.027	6.5
	Sum	0.042	25	0.17	7	0.18	17.5	0.16	20	0.26	11	0.042	17
MST	NB	0.011	9.5	0.24	21	0.13	13.5	0.12	13.5	0.35	28.5	0.042	17
	DecT	0.0092	4	0.15	3	0.089	4.5	0.058	5	0.26	11	0.036	13
	Prod	0.48	31.5	0.33	31.5	0.42	31.5	0.56	31.5	0.35	28.5	0.37	31.5
	Max	0.48	31.5	0.33	31.5	0.42	31.5	0.56	31.5	0.35	28.5	0.37	31.5
	Min	0.48	31.5	0.33	31.5	0.42	31.5	0.56	31.5	0.35	28.5	0.37	31.5
	Median	0.48	31.5	0.33	31.5	0.42	31.5	0.56	31.5	0.35	28.5	0.37	31.5
	Mean	0.0086	2.5	0.18	9.5	0.089	4.5	0.047	2	0.25	4.5	0.071	24
	W_Mean	0.0086	2.5	0.18	9.5	0.089	4.5	0.047	2	0.26	11	0.073	25
	Maj	0.011	9.5	0.21	18	0.13	13.5	0.047	2	0.28	17.5	0.053	23
	W_Maj	0.011	9.5	0.21	18	0.11	10	0.053	4	0.28	17.5	0.046	19
	Sum	0.011	9.5	0.16	5	0.087	2	0.12	13.5	0.27	15.5	0.038	14

TABLE 3.3: Results 1: Error rate of applying **RTC** classifiers based on the three neighborhood graphs RNG, GG and MST with respect to several aggregation functions

		Sat		Skin_NonSkin		Sonar		Spam		Threenorm		Twonorm	
		% Err.	Rank	% Err.	Rank	% Err.	Rank	% Err.	Rank	% Err.	Rank	% Err.	Rank
RNG	NB	0.16	22	0.0047	6	0.14	13.5	0.093	7	0.14	7	0.028	15
	DecT	0.11	6.5	0.0015	**1**	0.11	6	0.06	**1**	0.14	7	0.026	8.5
	Prod	0.46	28.5	0.034	23	0.46	27.5	0.29	27.5	0.4	27.5	0.29	27.5
	Max	0.26	26.5	0.034	23	0.46	27.5	0.29	27.5	0.4	27.5	0.29	27.5
	Min	0.46	28.5	0.034	23	0.46	27.5	0.29	27.5	0.4	27.5	0.29	27.5
	Median	0.26	26.5	0.034	23	0.46	27.5	0.29	27.5	0.4	27.5	0.29	27.5
	Mean	0.11	6.5	0.0043	5	0.085	1	0.075	3	0.14	7	0.027	11.5
	W_Mean	0.11	6.5	0.0065	9	0.12	10	0.078	5.5	0.14	7	0.027	11.5
	Maj	0.11	6.5	0.01	11	0.12	10	0.11	11.5	0.15	16	0.032	19
	W_Maj	0.11	6.5	0.013	14	0.11	6	0.11	11.5	0.15	16	0.03	18
	Sum	0.12	12	0.19	28	0.11	6	0.26	25	0.14	7	0.026	8.5
GG	NB	0.17	24.5	0.0052	8	0.24	18	0.096	8	0.14	7	0.023	4.5
	DecT	0.14	16	0.0019	3	0.2	15	0.062	2	0.13	1	0.023	4.5
	Prod	0.15	19	0.026	18.5	0.26	23	0.24	21.5	0.15	16	0.033	20.5
	Max	0.13	13.5	0.026	18.5	0.26	23	0.24	21.5	0.17	24	0.048	24
	Min	0.14	16	0.026	18.5	0.26	23	0.24	21.5	0.17	24	0.048	24
	Median	0.13	13.5	0.026	18.5	0.26	23	0.24	21.5	0.17	24	0.048	24
	Mean	0.15	19	0.005	7	0.24	18	0.077	4	0.14	7	0.024	7
	W_Mean	0.14	16	0.0069	10	0.24	18	0.078	5.5	0.14	7	0.023	4.5
	Maj	0.16	22	0.011	12.5	0.26	23	0.099	10	0.14	7	0.023	4.5
	W_Maj	0.16	22	0.011	12.5	0.25	20	0.097	9	0.14	7	0.022	2
	Sum	0.15	19	0.17	27	0.22	16	0.22	18.5	0.14	7	0.021	1
MST	NB	0.17	24.5	0.0033	4	0.14	13.5	0.19	15	0.16	21	0.029	17
	DecT	0.11	6.5	0.0016	2	0.1	3.5	0.13	13	0.15	16	0.028	15
	Prod	0.64	32.5	0.72	31.5	0.48	31.5	0.33	31.5	0.5	31.5	0.48	31.5
	Max	0.5	30.5	0.72	31.5	0.48	31.5	0.33	31.5	0.5	31.5	0.48	31.5
	Min	0.64	32.5	0.72	31.5	0.48	31.5	0.33	31.5	0.5	31.5	0.48	31.5
	Median	0.5	30.5	0.72	31.5	0.48	31.5	0.33	31.5	0.5	31.5	0.48	31.5
	Mean	0.11	6.5	0.034	23	0.095	2	0.22	18.5	0.15	16	0.027	11.5
	W_Mean	0.1	1	0.016	15	0.1	3.5	0.2	16.5	0.15	16	0.028	15
	Maj	0.11	6.5	0.037	26	0.12	10	0.2	16.5	0.16	21	0.034	22
	W_Maj	0.11	6.5	0.018	16	0.12	10	0.18	14	0.16	21	0.033	20.5
	Sum	0.11	6.5	0.51	29	0.12	10	0.25	24	0.15	16	0.027	11.5

		Waveform		Wine		Wisc_breast_cancer		Yeast	
		% Err.	Rank	% Err.	Rank	% Err.	Rank	% Err.	Rank
RNG	NB	0.15	9.5	0.49	28	0.035	17.5	0.57	21
	DecT	0.16	13	0.018	10.5	0.025	2	0.4	3
	Prod	0.53	28.5	0.49	28	0.071	27.5	0.69	30.5
	Max	0.41	26.5	0.37	23.5	0.071	27.5	0.61	24
	Min	0.53	28.5	0.49	28	0.071	27.5	0.69	30.5
	Median	0.41	26.5	0.37	23.5	0.071	27.5	0.61	24
	Mean	0.16	13	0.012	5.5	0.028	9.5	0.41	7
	W_Mean	0.15	9.5	0.018	10.5	0.029	12	0.41	7
	Maj	0.17	17.5	0.0059	2	0.026	5.5	0.47	16
	W_Maj	0.16	13	0.012	5.5	0.026	5.5	0.45	14
	Sum	0.15	9.5	0.029	17	0.037	19	0.61	24
GG	NB	0.14	4	0.45	26	0.044	21	0.52	18
	DecT	0.14	4	0.012	5.5	0.031	14	0.4	3
	Prod	0.15	9.5	0.21	21.5	0.049	23.5	0.66	26.5
	Max	0.19	23.5	0.12	19.5	0.049	23.5	0.47	16
	Min	0.19	23.5	0.21	21.5	0.049	23.5	0.66	26.5
	Median	0.19	23.5	0.12	19.5	0.049	23.5	0.47	16
	Mean	0.14	4	0.018	10.5	0.028	9.5	0.4	3
	W_Mean	0.14	4	0.018	10.5	0.028	9.5	0.4	3
	Maj	0.14	4	0.029	17	0.035	17.5	0.44	12
	W_Maj	0.14	4	0.029	17	0.031	14	0.41	7
	Sum	0.14	4	0.024	14.5	0.031	14	0.55	19.5
MST	NB	0.17	17.5	0.44	25	0.038	20	0.6	22
	DecT	0.17	17.5	0.018	10.5	0.025	2	0.4	3
	Prod	0.67	32.5	0.62	32.5	0.32	31.5	0.69	30.5
	Max	0.62	30.5	0.55	30.5	0.32	31.5	0.69	30.5
	Min	0.67	32.5	0.62	32.5	0.32	31.5	0.69	30.5
	Median	0.62	30.5	0.55	30.5	0.32	31.5	0.69	30.5
	Mean	0.17	17.5	0.018	10.5	0.028	9.5	0.42	9
	W_Mean	0.17	17.5	0.012	5.5	0.026	5.5	0.43	10
	Maj	0.19	23.5	0.0059	2	0.025	2	0.44	12
	W_Maj	0.18	21	0.0059	2	0.026	5.5	0.44	12
	Sum	0.17	17.5	0.024	14.5	0.034	16	0.55	19.5

TABLE 3.4: Results 2: Error rate of applying **RTC** classifiers based on the three neighborhood graphs RNG, GG and MST with respect to several aggregation functions

	RNG										
	NB	DecT	Prod	Max	Min	Median	Mean	W_Mean	Maj	W_Maj	Sum
Error_Mean	0.23	**0.12**	0.37	0.33	0.37	0.33	0.12	0.12	0.13	0.13	0.17
Rank	22.5	**2.5**	28.5	26.5	28.5	26.5	2.5	2.5	7	7	16
Rank_Mean	16.6	**5.68**	27.8	26.2	27.8	26.2	7.48	7.75	10.6	9.41	14.2
Rank	19	**1**	28.5	26.5	28.5	26.5	2	3	9	7	17

	GG										
	NB	DecT	Prod	Max	Min	Median	Mean	W_Mean	Maj	W_Maj	Sum
Error_Mean	0.24	0.13	0.23	0.21	0.23	0.21	0.15	0.15	0.16	0.15	0.18
Rank	25	7	22.5	19.5	22.5	19.5	12.5	12.5	15	12.5	17.5
Rank_Mean	17.5	7.86	20.3	20.5	21.9	20.5	10.8	10.8	13.6	12.3	15.2
Rank	20	5	22	23.5	25	23.5	10.5	10.5	14	12.5	18

	MST										
	NB	DecT	Prod	Max	Min	Median	Mean	W_Mean	Maj	W_Maj	Sum
Error_Mean	0.23	0.12	0.51	0.49	0.51	0.49	0.13	0.13	0.15	0.14	0.18
Rank	22.5	2.5	32.5	30.5	32.5	30.5	7	7	12.5	10	17.5
Rank_Mean	18.2	7.82	31.5	30.9	31.5	30.9	9.98	9.27	13.9	12.3	13.7
Rank	21	4	32.5	30.5	32.5	30.5	8	6	16	12.5	15

TABLE 3.5: The Mean Error and the Mean Rank for all the different **RTC** based of neighborhood graphs compared in tables 3.3 and 3.4

So, this means computing the distance $d(\omega_1, \omega_2)$ for each possible pair ($O(p^2)$ each for MAHALANOBIS) and compare the distance $d(\omega_1, \omega 2)$ with all distance $d(\omega_1, \omega)$ dans $d(\omega_2, \omega)$ for each individuals ω.

But optimization can be carried out :

- Distance can be computed using matrix representation and a powerful linear algrebra library (BLAS).
- Using the RNG condition, it is only necessary to test 3.2 for all ω such that $d(\omega_1, \omega) \leq d(\omega_1, \omega_2)$.
- For a given individual ω_1, sorting all distance $d(\omega_1, \omega)$ by increasing order. Then, we test the condition 3.2 following this order. If the distance $d(\omega_1, \omega_2)$ is large compared to the others, we reject the edge between ω_1 and ω_2 faster.

Practically, computing the results using RNG graphs is several times slower than using k-NN. For example with k=3, and the data set Twonorm (almost 2000 individuals), it takes 40s for k-NN method and 1min44s for the RNG method to compute N=100 classifiers. These tests use the BLAS library `atlas`, on a Intel™ Core™ i5 2.60GHz computer with 4G memory.

3.4.5 The choice of the appropriate statistical test

Choosing the appropriate statistical test to validate or compare algorithms is considered as one of the major Machine Learning problem. Dietterich [32] explains this problem in

	Arcene_expurge				Bupa				Glass			
	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank
RTC$_{RNG}$	0.215	-	-	3	0.3029	-	-	2	0.2095	-	-	1.5
RF	0.18	0.2091	0.2701	1	0.2853	0.2393	0.1718	1	0.2095	1	1	1.5
KNN1	0.225	0.6783	0.675	4.5	0.3823	0.0166	0.03	10	0.3	0.0101	0.0243	4
KNN3	0.21	0.8723	0.7995	2	0.3412	0.0019	0.0137	7	0.2857	0.0224	0.0209	3
KNN10	0.24	0.5291	0.4951	6	0.35	0.0258	0.0215	8	0.3524	0.0023	0.0144	7
SVM	0.225	0.6618	0.9324	4.5	0.3206	0.2789	0.2462	3	0.3239	0.0003	0.0059	6
AdaBoost	0.29	0.0714	0.0748	8	0.3235	0.4972	0.5933	4	0.5524	3.14E-05	0.0058	9
DA	na	na	na	na	0.3235	0.3633	0.3574	5	na	na	na	na
LR	0.54	9.59E-05	0.0058	9	0.3559	0.0676	0.0903	9	0.3762	4.40E-05	0.0058	8
C4.5	0.265	0.2368	0.3075	7	0.3382	0.234	0.22	6	0.3095	0.025	0.033	5

	Ionosphere				Iris				Letter_RvsB			
	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank
RTC$_{RNG}$	0.0543	-	-	2	0.0467	-	-	4.5	0.0097	-	-	1
RF	0.0714	0.1404	0.3598	3	0.0467	na	na	4.5	0.0164	0.0319	0.0418	5
KNN1	0.14	0.0078	0.0058	7.5	0.053	0.591	1	7.5	0.0145	0.1323	0.1275	3
KNN3	0.163	0.0034	0.0057	10	0.053	0.591	1	7.5	0.0131	0.343	0.3991	2
KNN10	0.163	0.00072	0.009	9	0.0467	0.999	0.8501	3	0.0191	0.0067	0.0215	6
SVM	0.0514	0.78	0.6845	1	0.0533	0.591	1	7.5	0.0151	0.137	0.1493	4
AdaBoost	0.0743	0.3983	0.235	4	0.033	0.1678	0.3458	2	0.0849	1.17E-07	0.0058	10
DA	0.131	0.0009	0.0087	6	na	na	na	na	0.0671	3.57E-06	0.006	9
LR	0.14	0.0004	0.0091	7.5	0.02	0.0368	0.0719	1	0.0618	2.33E-05	0.0058	8
C4.5	0.0971	0.00911	0.022	5	0.0533	0.591	1	7.5	0.0493	0.0002	0.0059	7

	Musk1				Parkinsons				Pima			
	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank
RTC$_{RNG}$	0.0788	-	-	1	0.0684	-	-	1	0.2526	-	-	7
RF	0.1	0.1066	0.1533	4	0.0947	0.1382	0.1041	4	0.229	0.0266	0.034	1
KNN1	0.1064	0.1461	0.2858	5	0.0737	0.7976	1	2	0.3013	0.037	0.0659	10
KNN3	0.0979	0.3305	0.3586	2	0.0895	0.1678	0.1759	3	0.2684	0.4151	0.362	9
KNN10	0.1404	0.018	0.0206	6	0.1	0.1678	0.1365	5	0.2513	0.9352	0.7667	6
SVM	0.0979	0.262	0.4185	3	0.1158	0.0676	0.0418	6	0.2368	0.013	0.0202	3.5
AdaBoost	0.2468	9.35E-06	0.0058	10	0.1421	0.0067	0.014	8	0.2605	0.4679	0.833	8
DA	0.1915	1.54E-05	0.0058	8	0.1263	0.02425	0.036	7	0.2368	0.1475	0.1219	3.5
LR	0.1957	1.19E-05	0.0059	9	0.1579	0.022	0.04967	10	0.234	0.1727	0.0759	2
C4.5	0.164	0.0009	0.011	7	0.1526	0.0368	0.0498	9	0.2487	0.7517	0.4401	5

	Ringnorm				Sonar				Spam			
	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank
RTC$_{RNG}$	0.0292	-	-	2	0.09	-	-	1	0.063	-	-	2
RF	0.0514	0.0039	0.0217	3	0.165	0.0522	0.0649	3	0.0456	0.0001	0.00589	1
KNN1	0.3302	1.02E-09	0.0059	6	0.165	0.0617	0.1222	3	0.0848	0.0019	0.011	6
KNN3	0.3873	6.34E-10	0.0059	8	0.165	0.043	0.0494	3	0.09	0.0001	0.0059	7
KNN10	0.4538	3.28E-10	0.006	9	0.29	0.0004	0.006	10	0.0948	0.0003	0.006	8
SVM	0.017	0.0049	0.014	1	0.18	0.029	0.075	0.0663	0.264	0.221	3	
AdaBoost	0.2424	8.02E-08	0.006	5	0.275	0.0001	0.0059	8.5	0.0985	1.38E-05	0.0059	9
DA	0.378	1.60E-10	0.006	7	0.26	0.001	0.013	7	0.113	3.80E-06	0.006	10
LR	0.4801	2.84E-11	0.006	10	0.275	0.0005	0.0091	8.5	0.083	0.0002	0.008	5
C4.5	0.1354	5.28E-06	0.006	4	0.22	0.0023	0.0141	6	0.0693	0.175	0.1548	4

	Threenorm				Twonorm				Waveform			
	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank
RTC$_{RNG}$	0.143	-	-	2	0.026	-	-	4	0.1548	-	-	4
RF	0.1439	0.832	0.9185	3	0.0307	0.032	0.0245	6	0.148	0.2722	0.386	3
KNN1	0.223	5.62E-05	0.0059	8	0.067	4.00E-06	0.0059	8	0.238	2.43E-07	0.006	7
KNN3	0.192	6.29E-05	0.006	0.045	0.001	0.0079	7	0.205	7.63E-07	0.006	6	
KNN10	0.154	0.0168	0.0355	4	0.0297	0.1527	0.1829	5	0.1704	0.0045	0.0059	5
SVM	0.128	0.0176	0.025	1	0.0212	0.096	0.1411	2	0.136	0.0005	0.006	1
AdaBoost	0.32	2.62E-08	0.006	10	0.14	1.20E-08	0.0059	9	0.2982	1.16E-06	0.0059	9
DA	0.161	0.028	0.0433	5	0.0193	0.0248	0.0247	1	na	na	na	na
LR	0.1674	0.0056	0.011	6	0.023	0.2712	0.2684	3	0.1388	0.011	0.033	2
C4.5	0.279	2.04E-07	0.006	9	0.171	4.30E-07	0.0058	10	0.2384	6.11E-06	0.006	8

	Wine				Wisc_cancer				Yeast			
	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank	% Err.	t_pv	Wilcox_pv	Rank
RTC$_{RNG}$	0.0176	-	-	1.5	0.0265	-	-	2	0.39	-	-	2
RF	0.023	0.591	0.773	4	0.025	0.758	0.8334	1	0.3723	0.0117	0.0165	1
KNN1	0.047	0.0522	0.089	7	0.0382	0.011	0.01644	8	0.469	0.0001	0.006	8
KNN3	0.041	0.104	0.1736	6	0.0353	0.1679	0.1712	5	0.4534	0.0005	0.006	7
KNN10	0.0235	0.3434	1	4	0.031	0.2788	0.071	4	0.4196	0.0435	0.05	5
SVM	0.0177	1	1	1.5	0.0368	0.111	0.0743	6	0.4027	0.3687	0.359	3
AdaBoost	0.094	0.0019	0.0129	9	0.0485	0.0017	0.0085	10	0.5945	5.19E-07	0.0059	9
DA	na	na	na	na	0.0382	0.1039	0.1824	7	na	na	na	na
LR	0.0235	0.6783	1	4	0.031	0.4344	0.2041	3	0.4142	0.007	0.0128	4
C4.5	0.071	0.0414	0.058	8	0.044	0.0368	0.0343	9	0.4331	0.0077	0.02077	6

TABLE 3.6: Comparison of **RTC**$_{RNG}$ to several classification algorithms

	Error_Mean	Rank	Rank_Mean	Rank
$\mathbf{RTC}_{RNG}$	0.121037594	1	2.42	1
RF	0.124359279	2	2.78	2
KNN1	0.181045645	5	6.36	7
KNN3	0.174217321	4	5.64	4
KNN10	0.184980547	6	6.11	6
SVM	0.135873523	3	3.44	3
AdaBoost	0.22876118	9	7.86	9
DA	na	na	na	na
LR	0.206515274	8	6.06	5
C4.5	0.185481231	7	6.81	8

TABLE 3.7: Mean Error and Mean Rank of all the methods used in table 3.6

his work and creates a taxonomy of nine different cases in Machine Learning that need statistical validation. The subject of our work corresponds to the eight case : `Given two algorithms A and B and a small data set S, which algorithm will produce more accurate classifiers when trained on data sets of the same size as S?`. We reformulate this case as follow:

$\mathbf{H_0}$: No significant difference between algorithm A and B

$\mathbf{H_1}$: The two algorithms A and B are significantly different from each other

To test this case, Dietterich proposes five different statistical tests: McNemar's test, difference of propotions test, resampled paired t test, $k-$fold cross validated t test and 5×2cross validated t test. He recommends two tests:

- The McNemar's test if the test's execution can be done one time
- The $5 \times 2cv$ t test if the the test's execution can be done more than one time (ten times)

The recommendation of Dietterich hides the main reason of why this case is difficult to test. The samples of any sampling procedure like cross validation are not exactly independent. This samples are somehow related because it is generated by the same dataset. Therefore, the Student t test in this case, even if it has very high probability of type I error, cannot be trusted because it contradicts the main condition of using Student t test: the samples should be independent.

The $5 \times 2cv$ t test (see section 2.8.2 in chapter 2) is proposed as a good canditate because it has the highest power (Ability to reject $\mathbf{H_0}$ when it is false) and it directly measures variation due to the choice of training set. But, Alpaydin [1] prouves in his work that $5 \times 2cv$ t test is not effective because the test results may vary if we change the order of draw the folds (replications). To solve this problem, Alpaydin proposes the $5 \times 2cv$

Friedman test that has a lower risk of type I error and a higher power than the $5 \times 2cv$ t test.

Moreover, Demsar [29] proposes to solve our case by a non parametric statistical test, the Wilcoxon signed ranks test (see section 2.8.1 in chapter 2). He recommends Wilcoxon signed ranks test because it is safer than parametric tests. Parametric tests depend usually of many assumptions. If these assumptions are violated, a paramatric test like the Student t test became less powerful.

In the evaluation subsection 3.4.3, we used two statistical tests:

- The first one is the $10-$`fold cv` t test to ensure the equivalence between our method and Random Forests. And as Dietterich said in [32], if t test doesn't exhibit a difference between algorithms, then we can be confident that there is no difference.
- The second one is `Wilcoxon signed ranks` test [29]. It is less powerful than $10-$`fold cv` t test but it is safer as a non parametric test.

3.4.6 Why Delaunay is difficult to use ?

In this chapter, we introduce four different kind of neighborhood graphs, RNG, GG, MST and Delaunay Triangulation (DT) graph. We use all these graphs to create our classifier **RTC** except the DT graph because it is difficult to apply this graph in high dimensional space [3]. Delaunay graph is known as the biggest graph that contains all the other neighborhood graphs (see equation 3.1). Hence, This relation is used to explain the DT graph as the superset of GG, RNG, MST and NN graphs. This paper [3] shows that Gabriel graph can be used instead of Delaunay graph when it is not tractable in high-dimensional spaces.

In the other hand, to justify that it is not convenient to use Delaunay graph in our evaluation, we carry out a small experiment. We are counting the number of neighbors find out by each of the neighborhood graphs used in section 3.4.1. In this experiment, we use 23 different quantitative data sets from *UCI* Irvine repository. We apply MST, RNG and GG graphs to find the mean number of neighbors of any individual in a dataset. We calculate also the RCEW index [168] to evaluate the perforformance of each graph. The table 3.8 shows that GG graph has the highest number of neighbors and MST graph has the smallest number of neighbors [1]. We conclude that the number of neighbors is

[1] In table 3.8, We find that the mean number of neighbors for MST is 1.99 for all the datasets. It is easy to see that MST can have the mean of $\frac{2(n-1)}{n}$ neighbors where n is the number of individual of a dataset. This result can be proved by the recurrence relation.

	NbInd	Nbclass	NbVar	MST mean (RCEW)	RNG mean (RCEW)	GG mean (RCEW)
Arcene_expurge	200	2	1017	1.99 (0.251)	2.58 (0.259)	193.28 (0.493)
Glass	214	6	9	1.99 (0.323)	2.57 (0.356)	10.09 (0.491)
Ecoli	336	8	7	1.99 (0.202)	2.93 (0.223)	10.45 (0.287)
Transfusion	748	2	4	1.99 (0.325)	12.04 (0.279)	18.22 (0.293)
Sonar	208	2	60	1.99 (0.159)	2.80 (0.226)	98.46 (0.465)
Wine	178	3	13	1.98 (0.062)	3.15 (0.092)	28.65 (0.254)
Wisc_breast_cancer	681	2	9	1.99 (0.038)	22.19 (0.006)	61.59 (0.029)
Iris	150	3	4	1.98 (0.067)	2.52 (0.089)	5.97 (0.156)
Image_segmentation_total	2309	7	18	1.99 (0.051)	3.52 (0.082)	18.07 (0.239)
Ionosphere	351	2	33	1.99 (0.142)	2.29 (0.136)	25.31 (0.226)
Pima_for_graphes	768	2	8	1.99 (0.303)	3.68 (0.310)	31.45 (0.343)
Letter_RvsB_graphes	1524	2	16	1.99 (0.024)	3.60 (0.074)	50.88 (0.288)
Musk1	476	2	166	1.99 (0.128)	3.40 (0.166)	127.73 (0.470)
Bupa	345	2	5	1.99 (0.389)	3.07 (0.392)	12.60 (0.419)
Parkinsons	195	2	22	1.98 (0.087)	2.88 (0.120)	18.8 (0.218)
Ringnorm_ml	2128	2	20	1.99 (0.337)	4.22 (0.193)	370.94 (0.264)
Yeast	1484	10	8	1.99 (0.489)	3.78 (0.524)	24.44 (0.567)
Twonorm_ml	2128	2	20	1.99 (0.062)	6.30 (0.094)	574.35 (0.260)
Threenorm_ml	2128	2	20	1.99 (0.232)	6.32 (0.258)	601.46 (0.406)
Spam_base	4601	2	57	1.99 (0.103)	4.85 (0.082)	107.22 (0.289)
Waveform	5000	3	21	1.99 (0.234)	6.36 (0.250)	620.94 (0.391)
Cardiotocography_Normal_Other	2126	2	30	1.99 (1)	2125 (0.0009)	2125 (0.0009)
Internet_Ads_NA_complete	3279	2	1558	1.99 (0.074)	11.80 (0.035)	269.40 (0.170)

TABLE 3.8: Mean of the number of neighbors and the RCEW index find out by MST, RNG and GG graphs

increasing with respect to the volume of the graph represented in the following relation:

$$MST \subseteq RNG \subseteq GG \subseteq DT$$

Thus, DT might have, with strong probability, more neighbors than GG. Indeed, RCEW index shows the bad performance of GG graph in comparison to RNG graph. So, DT graph can be worse. These results show that Delaunay graph makes our study complicated. Therefore, DT graph in our case is difficult to use.

3.5 Conclusion

In this chapter, we have provided a new approach, the **Random Neighborhood Classification**, for using neighborhood structures in Ensemble Methods. The results obtained show that they are challenging the most powerful techniques such as Random Forests and SVM. Methods based on geometrical neighborhood graphs outperform the classic methods such as kNN. The advantage of **RTC** is the directly use of the neighbourhood graphs which allow us to use proximity matrix when the representation space is not explicit. The results seem interesting because they lead to many paths that can be taken to improve the performance of **RTC**.

We developped a library that contains all the functionalities of these experiments. This library has created in order to improve the performance of our classifier and compare it to many other method known in this field using the **R** functions. Indeed, using this library, we think that we can improve the results of **RTC** classifier by varying some parameters like the distance used or the sampling method.

In the following chapter, we apply our **RTC** classifier based on RNG to a real data extracted from **Twitter**. This work is in progress and the first results are good enough to encourage the continuity of this study (see chapter 5).

Chapter 4

Detection of Political Opinion in Tweets

4.1 Introduction

People today express their opinion freely and publicly using social networks like **Twitter**. Twitter is a micro blogging social network that started in 2006 and rapidly grew to more than 500 million users in 2012. Every day 140 million active users visit Twitter making almost 340 million tweets[1].

A tweet is a short message formed of no more than 140 characters. It can be a short sentence or phrase, a link to a website or a direct message to other users. Every user on Twitter can tweet, reply and retweet. They can also add hashtags to mark the topic of the tweet. All tweets of all users worldwide are public. Topics can be anything from product information to political news. Analysing tweets is central to opinion mining [8, 76, 119].

In the United States, many analysts attributed Barack Obama's victory in the 2012 presidential elections to the large and successful use of social media, including Twitter [2]. Twitter has become a prime destination for the latest trends and hot topics and has a big impact on modern politics.

In this work, we address the problem of detecting the political opinion of tweets. We propose a new approach to measure the political opinion using the tweet's author. Given a set of tweets written by a specific author about a specific politician, we try to identify

[1] http://en.wikipedia.org/wiki/Twitter

[2] http://www.whitehouse.gov/photos-and-video/video/2011/07/08/impressions-white-house-twitter-townhall

the opinion of that author towards that politician. More specifically, our approach consists on recognizing the political opinion expressed by a tweet based on classifying author's opinions rather than classifiying only the tweets.

To study this idea, we realize two distinct parts:

1. **Creating the Reference Set**: In the first part, we aggregate tweets for each author regarding to a specific politician and then decide its polarity or label (**With** or **Against** this politician)
2. **Label Generalization**: In the second part, we use the labeled author's tweets of the reference set as a learning sample in order to assign a label (**With**, **Against**) to other authors.

The first part is formed of two steps: manually label a small number of tweets expressed by each author about a specific politician. Then, we can apply the Naive Bayes aggregation method or the sampling method to find the label of each author regarding to a politician. In the second part, we use KNN and Random Forests methods to label (**With**/**Against**) all other authors.

This chapter is devoted to studying and validating these two parts. Therefore, we demonstrate that Bayes procedure and the sampling procedure can be good enough to label authors. And, we validate the KNN and Random Forests classification using leave one out procedure based on text analysis. The results show that our approach work well to find the author's opinion.

This chapter is organised as follows. Section 4.2 reviews some of the important work that had done in the field of tweet analysis. Section 4.3 states problems of analysing tweets and introduces notations, definitions and proposition useful to understand the process of this work. Section 4.4 presents our procedure details. Section 4.5 describes how we validate this procedure. Section 4.6 presents the conclusions of this study.

4.2 Related Work

The growth of Twitter, as a huge stream of real time data, has attracted researchers interested in analysing public opinion [65, 68, 85, 90, 114]. Therefore, some studies use Twitter to recommend real time news like the work of Phelan et al. [119]. Balasubramanyan et al. [5] try to show that analysing tweets may be a substitute to traditional polling. In this field, there are many similar studies to our own focused on sentiment analysis of texts using known approaches on Text Mining described in [55, 113, 115, 155, 158]

and the automatic classification of texts by topic [71] Sentiment analysis in social networks, in particular Twitter, has been done using a variety of approaches. Following, we represent these approaches in three different categories based on the way used to analyse tweets.

One body of work uses content, linguistic meaning of words and sentences: **Approaches based on tweet's text**. Some existing work of Twitter sentiment analysis based on content like the work of Maynard et al. [103] and Go et al. [56] involve building and using dictionaries capturing the sentiment typically associated with specific words. OpinionFinder identifies various aspects of subjectivity within sentences and sentiment expression [5]. Profile of Mood States (POMS) is a psychometric instrument used to evaluate the linguistic expression of emotions in tweets text [8]. Linguistic Inquiry and Word Count (LIWC2007) is a text analysis software developed to assess emotional, cognitive and structural components of text samples using a psychometrically validated internal dictionary [151, 161]. SentiWordNet [42] is one such resource, containing opinion information on terms extracted from the WordNet database and made publicly available for research purposes. Some works use Twitter to study satirical sentences [28]. Others [27, 56, 127] use emoticons and hashtags to classify tweets. And, other uses the trending topic [94] as new tool to analyse Twitter. In the following section 4.3, we explain why the approaches that focus on the text content of tweets are not effective.

Another body of work exploits the Twitter user's interest and network to analyse the tweets: **Approaches based on tweet's author**. Recently, many works show the importance of meta information to study tweets. Tan et al. [141] use relationship information from the Twitter follower/followee network to analyse user's comments. Lim et al. [98] present a framework for classifying the relative interests of Tweet's author using information from Wikipedia. Yamashita et al. [140] analyse and cluster sets of Twitter users based on the following relations. In the same way, we propose to use some reference tweet's authors that can help to study the opinion in Twitter.

Yet another body of work makes use of related topics and resources: **Approaches based on tweet's target**. Kinsella et al. [78] investigate the potential of improving topic classification in social media by using metadata retrieved from external hyperlinks in user generated posts. Pennacchiotti et al. [116] apply several semantic models to tweet categorization by matching tweets to their appropriate Wikipedia page. Other uses Wikipedia as training corpus [51, 105]. QI et al. [122] use neighborhood pages and Jiang et al. [69] use related tweets to improve the classification. Studying Tweets regarding a specific target seems to be easy and very helpful in this field. But, in the same time, finding the target of many tweets is difficult (see the examples described in section 4.3). Therefore, to avoid this problem, in the first time, we based on some

manually labeled tweet's author. At some point, this solution will help to limite the human intervention in the process of our approach .

On our knowledge, our approach seems to be the first that uses the author's opinions to detect the political opinion in Tweets and not only the tweet's content. This approach can be seen as a combination of all the three categories mentioned previously. Because in our approach, the second part uses the tweet's text to label other authors and the first part is based on the tweet's author regarding a tweet's target, a politician. However, our approach is obviously different and the following sections explain the reasons.

4.3 Problem Statement

Of course, to label an author, we need to analyse their tweets but classifying or labeling tweets is not easy. People may mention multiple topics in one tweet; they merge unrelated things in the same tweet... Indeed, statistical methods of text classification have difficulty with Twitter's short texts [118]. Tweets contain many meaningless words and short phrases. There are an enormuous number of these tweets that need time to extract, identify and analyse. At the same times, a lot of meta information can be useful like the follower, the followee, the "@" (person being addressed publicly), the "d" (person addressed privately), the "RT" (retweet, a repeat of another persons tweet), the "# "keywords (hashtags) and the trending topic (topic tagged many times rather than any other tags). Our goal is to find a workaround to solve the problem of Tweets opinion mining.

Most important work on analysing tweets today uses the sentiment analysis of words. However, mining opinion from tweets with such techniques is not efficient. Here are some examples that illustrate the challenges (see Appendix B.1):

1. `We regret the position of Francois Hollande, the retirement reform carried out by @NicolasSarkozy is necessary`

   This tweet contains two subjects @NicolasSarkozy and Francois Hollande. Using words sentiment score, this is a confusing case to find the class of the tweet because of the presence of two topics.

2. `It is unfair that someone who does not work or pay taxes can earn more than those who work !` *@NicolasSarkozy*

   In this example, words are more likely negative but the author actually supports the subject *@NicolasSarkozy*'s speech. Using most text mining methods, this tweet will be incorrectly labeled as having negative opinion. Using our

technique, we can find that the tweet's author speaks positively of the subject $@NicolasSarkozy$ based on others tweets written by the same author about the same subject.

3. `The delusions of grandeur of` $\#Flamby$ `will cost us a lot ! I'd rather vote for :`$@NicolasSarkozy$

   Here, the author who supports $@NicolasSarkozy$ is using a satiric name of $@fhollande$ ($\#Flamby$). In this case, it is difficult to assign the subject of the tweet.

The state of the art approaches for solving these problems are based on tweet's text (see section4.2). Many procedures were applied in this context to classify text as representing a positive or negative opinion on a topic. The examples described previously in this section and many others show how complicated it is to use methods based on content (sentiment words score) to label a political tweet. Our study yielded a new procedure based on labeling the author's opinion that can perform very well to address the problems described previously.

Following, some definitions that will help us to build the labeling procedure.

Definition 4.1. We define and explain, here, some notations and concepts:

- (**Subject**) Let $S = \{s_1, .., s_n\}$ be a set of n politicians that tweets talk about.
- (**Author**) Let $A = \{a_1, .., a_m\}$ be a set of m Authors.
- (Tweets contain subjects) For $s \in S$ Let $T_S(s)$ be the set of all tweets mentionning s in the text. This set is automatically computed. Subject s in $T_S(s)$ can be mentioned in tweet by itself or by their hashtag.
- (Author's tweets) For $a \in A$, $T_A(a)$ is the set of all tweets produced by a.
- (**Couple**) Let $C = A \times S$ the set of all possible couples of author and subject.
- (Couple's tweets) For a given couple $c = (a, s) \in C$ let $TC(c) = T_A(a) \cap T_S(s)$ the set of all tweets from the given author and mentionning a given subject.

Practically, a large part of a couple's tweet set is empty, so, we only consider a list of `interesting couples` which have more than 100 tweets. Let $c_1, \ldots, c_K \in C$ be the K `interesting couples`, we denote $TC_i = TC(c_i)$ the ith couple's tweet set. After we define a couple, we can resume the goal of this work by "labeling couples". To facilitiate this procedure, we define the following hypothesis:

Definition 4.2. Tweet of $t \in TC_i$ can be :

1. **Positive**: the author has a positive opinion of the subject or negative opinion of the subject's opponents, the set of such tweets is noted TC_i^+
2. **Negative**: the author has a negative opinion of the subject or positive opinion of the subject's opponents, the set of such tweets is noted TC_i^-
3. **Do not Know** (**CantSay**) or **Neutral**: the author has neutral or unsure opinion of the subject, the set of such tweets is noted $TC_i^\perp$

Proposition 4.3. *A couple c_i can be :*

1. ***With**: if more than 70% of $t \in TC_i$ are Positive. The set of "With" couple is denoted C^+.*
2. ***Against**: if more than 70% of $t \in TC_i$ are Negative. The set of "Against" couple is denoted C^-.*
3. ***Unspecified**: if not With nor Against. The set of such couple is denoted $C^\perp$.*

4.4 Approach Overview

In this section, we address labeling authors regarding subjects (couples). The input of our task is a collection of author's tweets extracted during the French election in 2012 regarding well known candidates. The output is the class label assigned to each couple. Figure 4.1 represents the main workflow which is framed in two parts:

1. **Creating the Reference Set**: We start by Data cleaning, extract *Subjects* and create the *Couples* set (*D_Couples*). Then, we apply the Bayes or sampling methods to determine the polarity of a set of tweets (square 1); (square 1);
2. **Label Generalization**: Using the Reference set of labeled authors (*D_ClassCouples*) as a learning sample, we assign a label to other authors (square 2);

These two phases are detailed below:

4.4.1 Creating the Reference Set

We first clean the data and extract all the subjects and authors. Then, we build *D_Couples* data formed of tweets of all the couples; $TC(c)$ $\forall c$. We use some of these couples called `interesting couples` based on some authors who wrote more than 100 tweets on a specific subject.

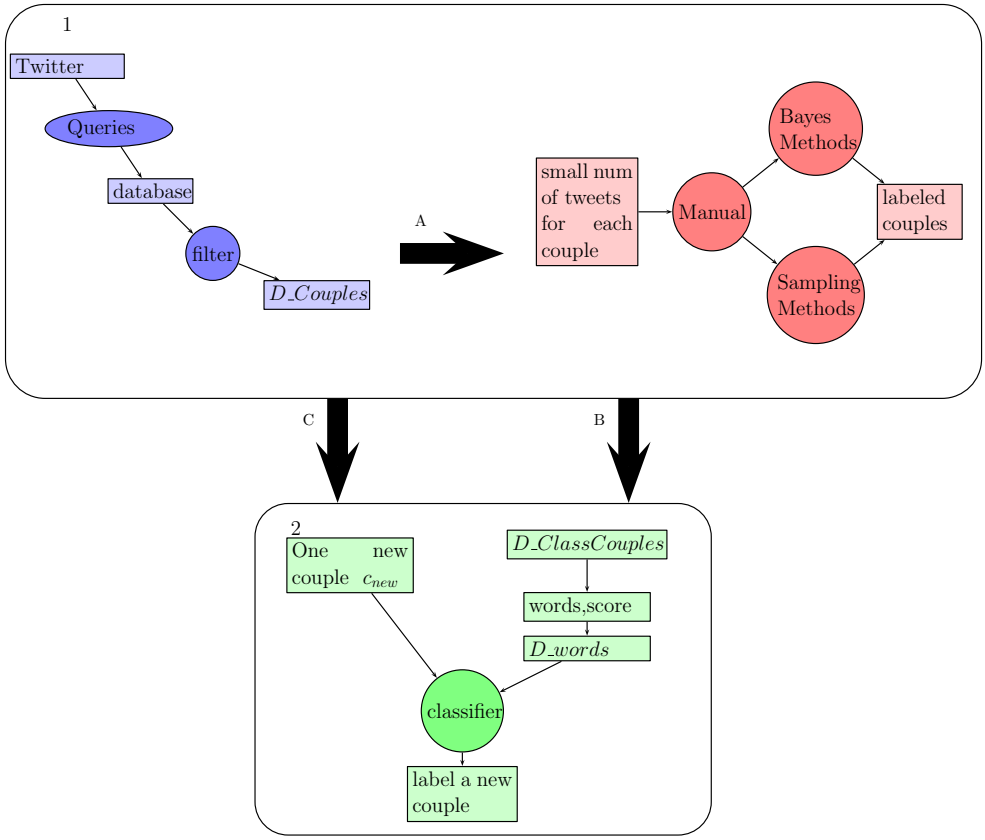

FIGURE 4.1: The two parts of our approach

Then, we try to find the tendencies of each couple which will help us to classify other tweet's authors later. Based on the assumption that using a small number of tweets (between 10 and 20 tweets) of a couple, we can deduce the label of this couple, we use methods that can be very effective in this case. So, to label a couple, we use one of these methods:

- The first is based on the Bayesian rule to calculate the probability of a couple to be **With, Against** or **Unspecified**.
- The second applies one of the acceptance sampling methods that can be used to estimate if the TC of a couple are Positive or not.

4.4.1.1 Bayes method

Naive Bayes (NB) is a popular aggregation method due to its computational efficiency and good predictive performance [48, 70, 163]. It is defined as a set of supervised learning algorithms based on applying Bayes Theorem with the naive strong independence assumptions. We use Naive Bayes method because it performs well with small samples of tweets to estimate the label of a couple.

Let a be an author and s be a politician where $c = (a, s)$ is the couple formed of this author and politician. We suppose that c has n tweets; $\left(TC^k\right)$ where $k = 1, \ldots, n$. We randomly extract 10 tweets among the n tweets of c. Then, we manually label these 10 tweets. Consider that we find the following labels:

$$(TC^k)_{k=1,\ldots,10} \text{ contains } \begin{cases} \alpha & \text{Positive tweets} \\ \beta & \text{CantSay tweets} \\ \gamma & \text{Negative tweets} \end{cases}$$

where $\alpha + \beta + \gamma = 10$

For each possible couple's label y_l ($y_1 =$ **With**, $y_2 =$ **Unspecified** , $y_3 =$ **Against**), we can estimate his probability using the Bayes theorem in the discrete form:

$$p(y_l/c) = \frac{p(y_l \cap c)}{p(c)} \tag{4.1}$$

$$= \frac{p(c/y_l) \times p(y_l)}{\sum_{i=1}^{3} p(c/y_i) \times p(y_i)} \tag{4.2}$$

where y_l can be one of the three different labels: **With** (y_1), **Unspecified** (y_2) and **Against** (y_3).

And because the label of all tweets of c are unkown, we create three profiles (see subsection 4.5.3) that characterize, with strong probability, the three labels of a couple (**With**, **Unspecified** and **Against**) based on Prop.4.3 as follow:

1. $P1$: profile of a supportive (**With**) couple TC^+: $(0.75, 0.2, 0.05)$ (which mean 75% of the TC are Positive, 20% are CantSay and 5% are Negative)
2. $P2$: profile of a **Unspecified** couple $TC^\perp$: $(1/3, 1/3, 1/3)$
3. $P3$: profile of a non supportive (**Against**) couple TC^-: $(0.05, 0.2, 0.75)$

Assuming that these profiles (TC_{ij}^+, $TC_{ij}^\perp$, TC_{ij}^-) are equiprobable. We choose arbitrary these profiles based on one assumption that represents very well the couple's labels.

However, the experiment (see section 4.5.3) shows that using real profiles measured from real observations do not change much the result.

Basically, we try to find whether the couple c is close to one of these profiles. So, using Bayes, we compute the probability that c follow one of those profils $P1$, $P2$, $P3$ considering small samples of tweets. The highest probability determines the label of the couple c. For exemple, if we get α positive, β cantSay and γ negative tweets, a small sample of $TC(c)$, the probability of c being close to y_1 is :

$$\begin{aligned} p(c \in y_1/c) \quad &= \\ &= \frac{p(c \in y_1) \times p(c/y_1)}{\sum_{i=1}^{3} p(c \in y_i) \times p(c/y_i)} \\ &= \frac{\frac{1}{3} \times p(c/y_1)}{\frac{1}{3} \times p(c/y_1) + \frac{1}{3} \times p(c/y_2) + \frac{1}{3} \times p(c/y_3)} \\ &= \frac{0.75^{\alpha} \times 0.2^{\beta} \times 0.05^{\gamma}}{(\frac{3}{4}^{\alpha} \times \frac{1}{5}^{\beta} \times \frac{1}{20}^{\gamma}) + (\frac{1}{3}^{\alpha} \times \frac{1}{3}^{\beta} \times \frac{1}{3}^{\gamma}) + (\frac{1}{20}^{\alpha} \times \frac{1}{5}^{\beta} \times \frac{3}{4}^{\gamma})} \end{aligned}$$

Example

Consider that we select randomly 10 tweets among all tweets of a couple c. Then, by manually classifying these tweets, we find 6 tweets Positive, 2 tweets CantSay and 2 tweets Negative .

Assuming that the 3 profiles of tweets are equiprobable: $p(c \in TC(c)^{+}) = p(c \in TC(c)^{\perp}) = p(c \in TC(c)^{-}) = 1/3$.

As the three profiles $P1, P2, P3$ form a Complete System of Events, we apply the Bayes' theorem [6, 20] as follow:

$$\begin{aligned} p(c \in y_1/c) & \\ &= \frac{p(c \in y_1) \times p(c/y_1)}{\sum_{i=1}^{3} p(c \in y_i) \times p(c/y_i)} \\ &= \frac{\frac{1}{3} \times p(c/y_1)}{\frac{1}{3} \times p(c/y_1) + \frac{1}{3} \times p(c/y_2) + \frac{1}{3} \times p(c/y_3)} \\ &= \frac{p(c/y_1)}{p(c/y_1) + p(c/y_2) + p(c/y_3)} \\ &= \frac{0.75^6 \times 0.2^2 \times 0.05^2}{(0.75^6 \times 0.2^2 \times 0.05^2) + (1/3^6 \times 1/3^2 \times 1/3^2) + (0.05^6 \times 0.2^2 \times 0.75^2)} \\ &= 0.51 \end{aligned}$$

We compute also $p(c \in y_2/c)$=0.49 and $p(c \in y_3/c) = 10^{-5}$;

Finally, the highest value determines the class label of $TC(c)$; here, the couple c is **With**.

4.4.1.2 Sampling procedures

In this section, we propose another method to detect the opinion of small sample of tweets, the sampling method. This method has a strong ability to predict opinion from a small sample. A sampling method, called also sampling inspection or Acceptance sampling, is used normally to determine whether to accept or reject the output (products or services) of large scale production line [110, 138, 152]. It is a tool of "decision making" regarding the acceptability of a lot. This method eliminates poor lots and improves the overall quality of product. Thus, we use this method to insure the reliability of our results.

In the literature, there exist many different sampling plans. Here, we describe some of the very known sampling plan as follow :

1. Single Sampling: Testing one sample, randomly selected, to decide.
2. Double Sampling: Testing a sample; if we can't decide, we use another sample to decide.
3. Multiple Sampling: Generalisation of Double Sampling; Each time a sample is inconclusive, we select another one.
4. Progressive Sampling: Same process as Multiple Sampling but using smaller sample.

We use the Progressive Sampling plan. It is based on testing many small samples; each time a sample is inconclusive, we select another one. Using, $D\_Couples$ and some TC manually classified, we proceed to classify each couple using progressive sampling plan (see figure 4.2). This method is severe in the way that classify tweets. It depend on the order that tweets are extracted. Here a simple example that explain the details of this method.

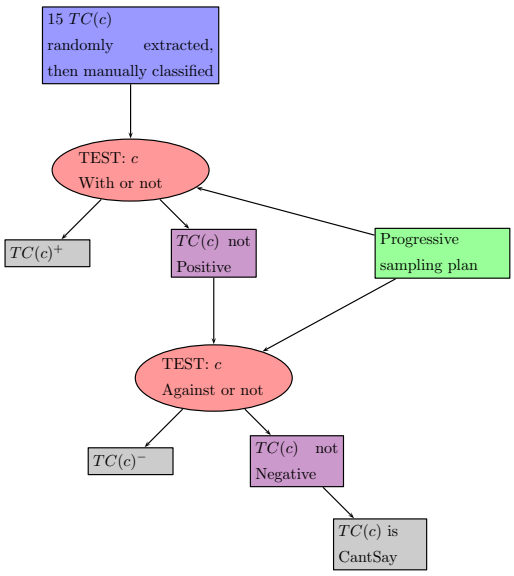

FIGURE 4.2: Strategy used in the Progressive sampling procedure

Using all tweets of a couple $TC(c)$, we start by selecting tweets randomly. Then, by testing whether the couple is **With** or not, we apply the progressive sampling plan using the table 4.1 [3]:

# of tested Tweets	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
acceptation								0	0	1	1	1	1	1	2
rejection	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3

TABLE 4.1: Progressive sampling plan

$TC(c)$ is positive if and only if:

1. all first eight tweets are Positive
2. among the first ten tweets extracted there is only one tweet that is not Positive
3. among the first five tweets extracted there are only two tweets that are not Positive

if $TC(c)$ is not Positive, we restart and test whether the couple is **Against** or not using the same table 4.1 and the same conditions. if $TC(c)$ is not Positive and not Negative $\Rightarrow$ $TC(c)$ is CantSay. This procedure is explained in figure 4.2.

[3] http://www.iutbayonne.univ-pau.fr/~grau/STID/cadrereception.html

4.4.2 Label Generalization

After we find the polarities of some couples (`interesting couples`) using Naive Bayes method in Section 4.4.1, we determine the opinion of other authors (see figure 4.3). The labeled `interesting couples` are used as a reference dataset in order to label all the remaining couples of $D\_ClassCouples$. The goal is to create a big labeled text corpus that will be the new reference to label any new author.

First, we extract and weight words of each TC^k of all the `interesting couples`. More specifically, we use the `treetag` function of the `korRpus` package [104], to extract all words of the labeled TC. Using TreeTagger that helps to define each word in a text, we create a database of words, noted D_words, formed of all words that `treetag` can extract from tweets. We filter and clean the data of words in D_words by removing punctuation and all the "subject" words.

Then, we use the **TF-IDF** Term Frequency Inverse Document Frequency to determine the frequency of each word in the corpus of all TC. TF-IDF is a classical weighting method used in information retrieval. It consists to find the important terms for the document within a corpus by assessing how often the word occurs within a document (TF) and how often in other documents (IDF).

Each word of each tweet of all the `interesting couples` has a weight that will help to label other authors. This labeling is given based on how much a new author close to one of the `interesting couples`.

Second, let c_{new} be a new couple and TC^k_{new} be the k tweets of this couple. Suppose that D_words is the database of all the weighted and labeled tweets of the `interesting couples`.

We extract and weight words of TC^k_{new} based on the same procedure used to weigh all the `interesting couples`. Then, we compare the weighted words of TC^k_{new} with the weighted and labeled words of D_words using a proximity measure. Finally, we find that c_{new} is close to one of the labeled `interesting couples` and this result decide to which label c_{new} will be affected. And then, we can label all the remaining couples in $D\_ClassCouples$.

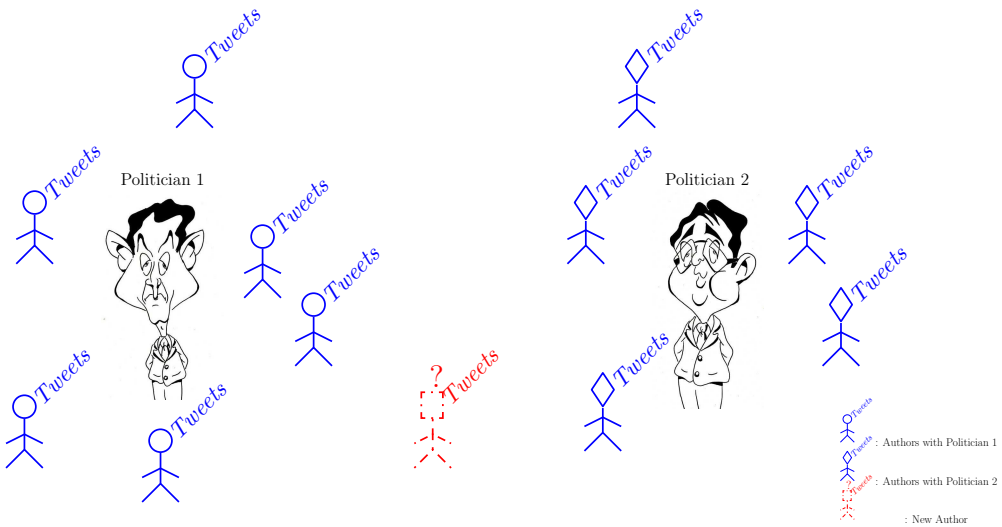

FIGURE 4.3: The goal of the Label Generalization is labeling a new tweet's author

4.5 Evaluation and Results

4.5.1 Corpus

In this study, we use 300,000 tweets published during the two months before the French elections in Spring 2012. This database contains 133 couples with more then 100 tweets for each couple. We define a subject as a candidate of this election. We create 2 databases. The first one is formed of couples using all subjects, in order to study the efficiency of the Bayesian and sampling estimation. In the second, we use two subjects, Francois Hollande and Nicolas Sarkozy, because many authors talk a lot about these two candidates and almost all authors are With or Against one of them. In the second database, authors can be: **With** (with Hollande or against Sarkozy) or **Against** (against Hollande or with Sarkozy).

	First database	Second database
Subjects	All candidate	F. Hollande/N. Sarkozy
# of couples	72	99
With	52	41
Against	10	58
$\perp$	10	-

TABLE 4.2: Description of the two databases used in this work

4.5.2 Validation of the Naive Bayes labeling procedure and the Sampling methods

In this validation, we try to prove that labeling a small number of tweets via Bayesian or Sampling procedure is effective.

Using *D_Couples* database, we select 72 couples and we manually label 50 of their tweets:

1. 52 couples are **With** (contains more than 75% of positive tweets);
2. 10 couples are **Against** (contains more than 75% of negative tweets);
3. 10 couples are $\perp$ (not **With** nor **Against**);

Then, we use Naive Bayes method and the sampling method to label the same 72 couples. By repeating the Naive Bayes labeling and the sampling method based on choosing randomly different 10 manually labeled tweets, we calculate the Accuracy percentage. Table 4.3 shows the results:

	Bayes Estimation		
Manual labeling	With	Unspecified	Against
With	98.99%	1.05%	0%
Unspecified	12.3%	74.98%	12.72%
Against	0%	3.27%	96.73%
	Sampling Estimation		
With	43%	57%	0%
Unspecified	0.3%	99.7%	0%
Against	0%	60%	40%

TABLE 4.3: Comparing the manual labels with Naive Bayes and sampling labels

We find out that, using Naive Bayes method, approximately 98% of **With** couple are labeled **With** and approximately 96% of **Against** couples are labeled **Against**. Indeed, using sampling plan, there are 43% of **With** couple are labeled**With** and 40% are of **Against** couples are labeled **Against**. The sampling plan are more severe on classifiying tweets (see paragraph 4.4.1.2). This table shows that our technique is effective to determine the opinion of a couple using a small number of their tweets.

4.5.3 Why these specific profiles?

In this work, we use the profiles described in subsection 4.4.1 that represent with strong probability the three couple's labels. However, there exist many different profiles that can be appropriate in this case. Why we choose this specific profiles?

To explain this choice, we build an experiment to test if there is any difference among three different cases of profiles: the one that we are using (profiles 1), another one choosed randomly (profiles 2) and a real one provided by real observations (profiles 3). These real observations are the data of the first database (see table 4.2) where we manually labeled 50 tweets of each one of the 72 couples. Table 4.4 describes the three cases of profiles. Then, we apply for each one of these three cases the same validation test used in the previous subsection 4.5.2. We find three confusion matrix. We calculate the Accuracy of each matrix (see table 4.5). The results show that the difference between the three Accuracy do not exceed the 3.5% . Which means that using one of the three appropriate cases do not change enormously the results. So, the profiles used to evaluate our approach are effective.

	Profiles 1		
With	0.75	0.2	0.05
Unspecified	0.33	0.34	0.33
Against	0.05	0.2	0.75
	Profiles 2		
With	0.80	0.15	0.05
Unspecified	0.33	0.34	0.33
Against	0.05	0.15	0.80
	Profiles 3		
With	0.87	0.09	0.04
Unspecified	0.23	0.37	0.40
Against	0.01	0.11	0.88

TABLE 4.4: Three different profiles

	Profiles 1	profiles 2	profiles 3
Accuracy	0.883	0.907	0.917

TABLE 4.5: The corresponding Accuracy of the three confusion matrix calculating using the three different profiles cases

4.5.4 Results

To assess the performance of our approach, we try to find the opinion of a new author (new couple). We want to validate whether the labeling procedure, described in Section 4.4.2, is reliable. We manually label 15 tweets of each of the 99 couples. Then, we apply our technique, described in Section 4.4, to label these couples (Tab.4.2). We extract and weight words using the labels of the 99 couples via Naive Bayes. The words that we find in the tweets talking about Hollande and Sarkozy are presented in the following figures:

- In figure 4.4, we can see all words used by the 58 authors **With** Sarkozy (**Against** Hollande).
- In figure 4.5, we can see all words used by the 41 authors **With** Hollande (**Against** Sarkozy).



FIGURE 4.4: Words of all tweets written by authors **With** Sarkozy (**Against** Hollande)

Then, we apply the leave one out procedure using two well known classifiers, K Nearest Neighbors (KNN), Random Forest (RF) and our classifier $\mathbf{RTC_{RNG}}$ described in chapter 3. All the calculations are done using the statistical software **R** (see CRAN.r). For KNN, we use the `knn` function of the `class` library and we compute KNN with k=1,2 and 3. The implementation of RF is the `randomForest` library using 500 trees. We adjust 2 parameters in `randomForest` function to adapt the data in the calculation. The first one is `mtry`, the number of possible splitting variables which are sampled at each node and the second one is the `cutoff` vector used in random forest to determine the classification. For $\mathbf{RTC_{RNG}}$, we used the Relative Neighborhood Graph (RNG) and the Decision Template aggregation function.

The results are reported by the confusion matrix in Tab.4.6. Also, we compute the Accuracy of the labeling for each classifier. These results show the good performance of our procedure with approximately 84% of Accuracy using KNN for k=3, 83 % using Random Forests and 71% using $\mathbf{RTC_{RNG}}$.



FIGURE 4.5: Words of all tweets written by authors **With** Hollande (**Against** Sarkozy)

	KNN1		KNN2		KNN3		RandomForest		RTC_{RNG}	
	With	Against	With	Against	With	Against	With	Against	With	Against
With	36	5	33	8	30	11	36	5	25	16
Against	29	29	28	30	5	53	10	48	12	46
Accuracy	0,6565657		0,6363636		0,8383838		0,8484848		0,7171717	

TABLE 4.6: Results of Leave one out procedure

4.6 Conclusion

In this chapter, we propose a new approach to detect the political opinion in tweets by using the author's opinion. This work is based on studying the tweet's author regarding political subjects (politicians) in order to mine political opinion in tweets. We label a new attribute called **couple**, formed of an author and a subject. We use a real data set to validate our approach.

This approach is formed of two parts. The first consists of creating a reference set of labeled couples based on a small number of their tweets. We discussed and proved the first one. The second part aims to label new authors in order to discover the supporters

of a politician and be able to count them. The results of the second part encourage further work.

We apply a second method called sampling plan to compare the results of the Naive Bayes method. The sampling plan method is severe in comparison to Naive Bayes. But, this method prove that Naive Bayes method is a good candidate to create the labeled reference set.

This chapter provides a real database experiments to apply our $\mathbf{RTC_{RNG}}$ classifier. We find that our method based on neighborhood graph (RNC) can be a good classification method in comparison to the most efficient method in this field, the Random Forests.

Chapter 5

Conclusion

We address here a review of the research work that we have done during this PHD and some principles perspectives for future work.

5.1 Thesis Review and Conclusions

In this thesis, we provide a review over the main approaches and algorithms used in Machine Learning to solve the classification problem of the supervised learning. These methods are described in chapter 2.

In chapter 3, we present an original new method, called **Random Neighborhood Classification (RNC)**. This method have been successfully validated on a number of data sets taken from the *UCI* Irvine repository and compared to many different methods such as Random Forests and Support Vector Machine. RNC method created based on two concepts:

1. The **topological learning**;
2. The **Ensemble Methods** procedure;

Topological learning is a framework for machine learning where the topology of the data set is the basic concept used to construct a classifier. We propose to use geometric graphs to build the topology such as Gabriel Graphs (GG) or Relative Neighborhood Graphs (RNG). These graphs provide a neighborhood structure that express the topological structure of the data set in multidimensional space. We encourage the topological type of learning because it can provide some advantages in the following issues :

- `Learnability` : How can the topological framework help us to address the learnability issue, which is a challenging question in machine learning?
  Many studies can explain why topological structure can be useful such as :
  - The statistic index RCEW, described in chapter 2 section 2.7, proposed by Zighed, D. A., Lallich, S. and Muhlenbach F. (2002) [168] that measures how classes can be disconnected based on the propotion of the cut edges constructed by a neighborhood graph;
  - The work proposed by Aupetit, M. and Catz, T. (2005) [3] about the topological graph and the information that can be extracted from it;
- `Improving the representation space` : How can we generate a better representation space where the classes overlap less?
  In Machine Learning, Ensemble Methods (EM), are popular due to their efficiency in prediction. EM use multiple classifiers generated in random space from a data set, then aggregate these classifiers into one. This aggregated classifier has leads to a better classification and shows that the topological structure changes advantageously [11, 17, 35, 83].

In chapter 4, we provide a real application, extracted from Twitter, to our **RNC** method. The choice of Twitter is based on one reason: Twitter is one of the most used and attractable social networks where a huge number of people express their opinion about several topic especially the political issues.

But, analysing tweets is not very easy, especially, that tweets texts are formed of short phrases and it contains link to website, emoticons and different symbols. For all these reasons and others explained in chapter 4, we propose a new procedure to analyse tweets that focus on the tweet's authors and not the tweet's text. We create a new attribute called **couple** that represents the input in this procedure. We try to find if an author who write about a candidate, a couple, is **With** or **Against** (outputs) this candidate.

The procedure described in chapter 4 is formed of two parts. The first one provides methods used to create the reference set of labeled couples. In this part, we propose two different methods, the Naive Bayes and the sampling plan procedure. The second part uses the reference set as a learning sample in order to classify new unlabeled authors. This part can be done using any classifier from the supervised learning. Thus, we apply our RTC classifier based on neighborhood graph and other classifiers, the RF and KNN.

5.2 Ideas for future work

The following section surveys ideas for future research to improve the topological study in the supervised learning field and future directions to dected opinion from tweets.

For topological study in the supervised learning field, there are the following additional developments that could further extend the usefulness of **Random Neighborhood Classification** results.

In chapter 3, we introduce our classification method **RNC** based on topological graph. The results find by applying **RNC** were very encouraging. We think that these results can be improved by changing the following parameters:

- The type of base classifier. Should we use the closest connected homogeneous component? How to define precisely this notion of proximity ? Should we take into consideration the size of the database or other characteristics of the neighborhood such as density, etc...?
- The selection methods of datasets variables. The ability to choose relevant variables can be very helpful because neighborhood graphs are sensitive to dimensionality and outliers.
- The choice of the dissimilarity measure between individuals in the geometrical graphs. We can use other distance instead of Mahalanobis like Minkowski distance.
- The determination of the membership class of an unclassified individual. We have applied a simple vote in the neighborhood of the individual to be assigned to one class. We can weight the vote of an individual in the learning set according to its distance to the unclassified one. We can also take into account, not only the direct neighbors, but also, their neighbors until a certain level. There exist plenty of assignment procedures based on homogeneous clusters and so forth.

For the particular area treated in chapter 4 about the Political Opinion in Tweets, the following directions offer interesting opportunities for future work that in some way can improve the procedure used to analyse political opinion in tweets:

First, the second part of the procedure used in chapter 4 can be improved by:

- Analysing the Hashtags and Emoticons; We can use one of the proposed procedure in the literature that deal with Hashtags and Emoticons like the work presented in these papers [27, 34]

- Studying the sarcasm problem; In this field of tweets (political tweets), authors can deride of any politician by using irony words or by writing entire tweets that has abstract meaning. Many works today are studying this problem like the semi supervised recognition of sarcastic sentences presented in paper [28].
- Using the followers and following; The detection of followers and following can help to improve labeling couples.
- Using other classifiers; We can apply other classifiers and compare between them in order to find the best classifier to use in this case.

These improvements can affect positively the opinion detection of tweets.

Second, we think that the procedure used in chapter 4 can be more effective if we suppose that all the work done in chapter 4 is the starting point or the core of a new way to analyse tweets. Then, we will need to repeat the same work to other period of time using political tweets. This work will help to analyse the opinion evolution and the popularity of politician in tweets.

Indeed, A descriptive presentation of this evolution can be very helpful for more understanding of the opinion progression. For example, we can construct the circle of supporters similar to that presented in figure 4.3 for each period and try to add more authors and more politicians. Then, this representation is repeated to other period of tweet's collecting.

Appendix A

Machine Learning

A.1 Evaluation measure

We recall here the definition of the very known evaluation measure, precision, recall and accuracy.

Test / Truth	positive	negative
positive	True positive T_p	False positive F_p
negative	False negative F_n	True negative T_n

TABLE A.1: Contingency table that represents the classification results of an experiment

$$\textbf{Accuracy} = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$\textbf{Precision} = \frac{T_p}{T_p + F_p}$$

$$\textbf{Recall} = \frac{T_p}{T_p + F_n}$$

A.2 Dataset Example

Ind	X_1	X_2	Y	Ind	X_1	X_2	Y
1	8.4875319	2.7396460	1	16	0.7700618	9.1903390	2
2	3.6651842	2.5244115	1	17	7.2221281	6.2204873	2
3	9.9315399	9.3000004	2	18	4.0727986	7.7520187	1
4	0.2314684	8.8392329	2	19	4.9639694	2.7941412	1
5	5.1644242	2.1798260	1	20	5.0563111	2.3069185	2
6	2.5496821	8.6987640	2	21	9.6544542	0.7431042	2
7	0.3424954	2.8353526	2	22	2.7008693	8.2018315	1
8	6.0092604	5.5848290	2	23	3.9142067	8.1987593	2
9	0.8824149	7.4892884	1	24	9.1562638	3.0781256	1
10	6.4786939	7.9792962	1	25	7.4951086	8.1023719	1
11	1.1588053	1.0912910	1	26	3.8135361	2.2621388	1
12	4.5784517	6.1216656	2	27	8.9639833	6.4018395	2
13	8.3047590	8.6454260	2	28	0.6051228	9.1386579	1
14	1.9084150	4.5581447	2	29	2.2246908	3.5463861	2
15	9.5135340	3.7362075	1	30	8.4979244	1.7475695	1

TABLE A.2: Dataset Example

A.3 Suitable statistical test

All machine learning algorithms are statistically tested in order to verify their performances. The statistical tests are an essential tool of decision making in favor or against an hypothesis studied. Their exist many statistical tests which are choosed based on four variables :

1. the type of features in the data: quantitative, ordinale, nominale
2. the distribution of the data sets studied: Gaussian (parametric) or non parametric
3. Comparing two sets or many sets
4. the sets are related or independant

The figure A.1 in page 97 describes how we can choose the efficient test based on the data description.

Statistical tests are :

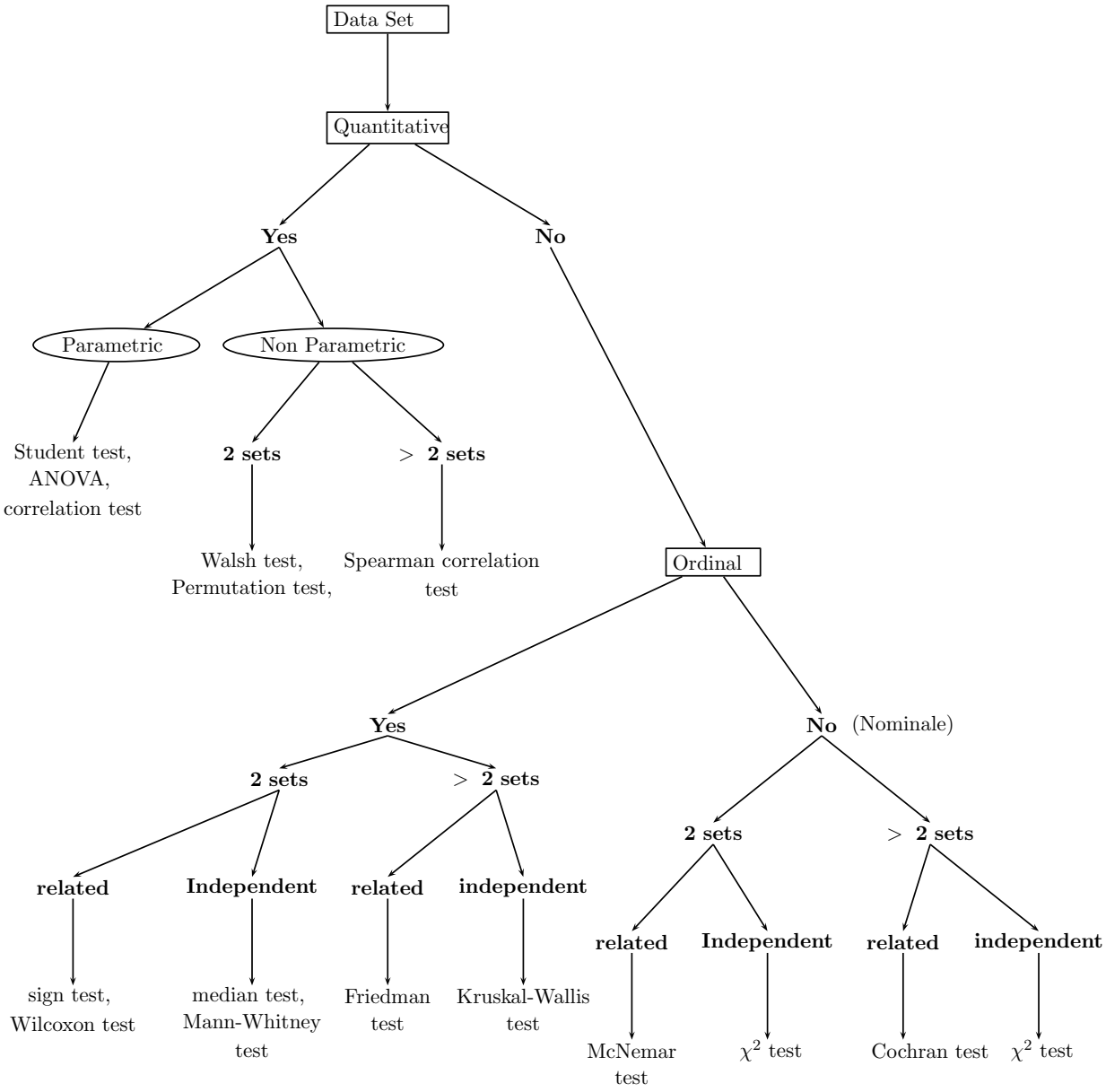

FIGURE A.1: Choice of statistical test based on the data description

1. Parametric: a statistic is called parametric if the probability distribution of the data is completly known, normally, following the Gaussian law.

2. Robust: the results found using the test is valide even if the data is not normally distributed.

3. Non parametric: test is valide whatever the distribution of the data. This test is used when the data is not belonging to any particular distribution.

The table A.3 in page 98 describes how to choose a statistical test based on the relationship between the dependent variables and the explanatory variables.

Exploratory Variables	Dependent Variables			
	Nominale 2 categories	**Nominale > 2 categories**	**Ordinale**	**Quantitative**
Nominale 2 categories				
Independent	test for the difference of proportion or χ^2	χ^2	Mann-Whitney	student test
Related	McNemar test	Cochran test	sign test	paired t test
Nominale > 2 categories				
Independent	χ^2	χ^2	test of Kruskal-Wallis	ANOVA
Related	Cochran test	Cochran test	Friedman test	two-way ANOVA
Quantitative	student test	ANOVA	Spearman correlation	Pearson Correlation

TABLE A.3: Statistical tests based on the type of the dependent and independent variables

A.4 Wilcoxon Test

Critical Values for the Wilcoxon Signed Ranks test

n	α1 = α2 =	5% 10%	2.5% 5%	1% 2%	0.5% 1%	n	α1 = α2 =	5% 10%	2.5% 5%	1% 2%	0.5% 1%
1		—	—	—	—	26		110	98	84	75
2		—	—	—	—	27		119	107	92	83
3		—	—	—	—	28		130	116	101	91
4		—	—	—	—	29		140	126	110	100
5		0	0	—	—	30		151	137	120	109
6		2	0	—	—	31		163	147	130	118
7		3	2	0	—	32		175	159	140	128
8		5	3	1	0	33		187	170	151	138
9		8	5	3	1	34		200	182	162	148
10		10	8	5	3	35		213	195	173	159
11		13	10	7	5	36		227	208	185	171
12		17	13	9	7	37		241	221	198	182
13		21	17	12	9	38		256	235	211	194
14		25	21	15	12	39		271	249	224	207
15		30	25	19	15	40		286	264	238	220
16		35	29	23	19	41		302	279	252	233
17		41	34	27	23	42		319	294	266	247
18		47	40	32	27	43		336	310	28	261
19		53	46	37	32	44		353	327	296	276
20		60	52	43	37	45		371	343	312	291
21		67	58	49	42	46		389	361	328	307
22		75	65	55	48	47		407	378	345	322
23		83	73	62	54	48		426	396	362	339
24		91	81	69	61	49		446	415	379	355
25		100	89	76	68	50		466	434	397	373

Appendix B

Detection of Political Opinion in Tweets

B.1 The original French Tweets Examples

- nous regrèttons la position de Francois Hollande, la réforme des retraités menée par @NicolasSarkozy est nécessaire.
  We regret the position of Francois Hollande, the retirement reform carried out by @NicolasSarkozy is necessary.

- C'est une injustice quand celui qui ne travaille pas et ne cotise pas peut gagner davantage que celui qui travaille! @$NicolasSarkozy$.
  It is unfair that someone who does not work or pay taxes can earn more than those who work ! @$NicolasSarkozy$.

- Les folies des grandeurs de #$Flamby$ vont nous coûter un bras ! Je vote plutôt : @$NicolasSarkozy$.
  The delusions of grandeur of #$Flamby$ will cost us a lot ! I'd rather vote for :@$NicolasSarkozy$.

Bibliography

[1] Alpaydin, E. Combined 5 x 2 cv f test for comparing supervised classification learning algorithms. *Neural Computation 11*, 8 (1999), 1885–1892.

[2] Athitsos, V., Alon, J., and Sclaroff, S. Efficient nearest neighbor classification using a cascade of approximate similarity measures. In *CVPR (1)* (2005), IEEE Computer Society, pp. 486–493.

[3] Aupetit, M., and Catz, T. High-dimensional labeled data analysis with topology representing graphs. *Neurocomputing* (2005), 139–169.

[4] Baert, A. *Graphes aléatoires et diffusion dans les réseaux cellulaires: modélisation combinatoire et probabiliste.* 2003.

[5] Balasubramanyan, R., Routledge, B. R., and Smith, N. A. From tweets to polls : Linking text sentiment to public opinion time series, 2010.

[6] Bernardo, J., and Smith, A. *Bayesian Theory.* Wiley Series in Probability and Statistics Series. John Wiley & Sons Canada, Limited, 2007.

[7] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[8] Bollen, J., Pepe, A., and Mao, H. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *CoRR abs/0911.1583* (2009).

[9] Breiman, L. Bagging predictors. In *Machine Learning* (1996), pp. 123–140.

[10] Breiman, L. Bias, variance, and arcing classifiers. Tech. rep., 1996.

[11] Breiman, L. Bias, variance, and arcing classifiers. *Statistics* (1996).

[12] Breiman, L. Some infinite theory for predictor ensembles, 2000.

[13] Breiman, L. Random forests. *Machine learning 45*, 1 (2001), 5–32.

[14] Breiman, L. Random forests. *Mach. Learn.* (2001), 5–32.

[15] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Chapman & Hall, New York, 1984.

[16] Brighton, H., and Mellish, C. Advances in instance selection for instance-based learning algorithms. *Data Min. Knowl. Discov.* (2002), 153–172.

[17] Brown, G., Wyatt, J., Harris, R., and Yao, X. Diversity creation methods: a survey and categorisation. *Information Fusion 6*, 1 (2005), 5–20.

[18] Burges, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* (1998), 121–167.

[19] Chazelle, B. A minimum spanning tree algorithm with inverse-ackermann type complexity. *J. ACM* (2000), 1028–1047.

[20] Congdon, P. *Bayesian statistical modelling*. Wiley series in probability and statistics. Wiley, 2001.

[21] Cortes, C., and Vapnik, V. Support-vector networks. In *Machine Learning* (1995), pp. 273–297.

[22] Cover, T., and Hart, P. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on* (1967), 21–27.

[23] Criminisi, A., Shotton, J., and Konukoglu, E. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. Tech. rep., Microsoft Research, 2011.

[24] Criminisi, A., Shotton, J., and Konukoglu, E. Decision forests for classification, regression, density estimation, manifold learning and Semi-Supervised learning. Tech. rep., Microsoft Research, 2011.

[25] Cristianini, N., Kandola, J., Elisseeff, A., and Shawe-Taylor, J. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14* (2002), MIT Press, pp. 367–373.

[26] Cunningham, P., and Delany, S. J. k-nearest neighbour classifiers. Tech. rep., 2007.

[27] Davidov, D., Tsur, O., and Rappoport, A. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters* (Stroudsburg, PA, USA, 2010), COLING '10, Association for Computational Linguistics, pp. 241–249.

[28] Davidov, D., Tsur, O., and Rappoport, A. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning* (Stroudsburg, PA, USA, 2010), CoNLL '10, Association for Computational Linguistics, pp. 107–116.

[29] Demsar, J. Statistical comparisons of classifiers over multiple data sets, 2006.

[30] Dietterich, T., Kearns, M., and Mansour, Y. Applying the weak learning framework to understand and improve c4.5. In *In Proceedings of the Thirteenth International Conference on Machine Learning* (1996), pp. 96–104.

[31] Dietterich, T. G. Machine-learning research – four current directions. *AI MAGAZINE* (1997), 97–136.

[32] Dietterich, T. G. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation* (1998), 1895–1923.

[33] Dietterich, T. G. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems* (London, UK, 2000), MCS '00, Springer-Verlag.

[34] Dolan Antenucci, Gregory Handy, A. M., and Tinkerhess, M. Classification of tweets via clustering of hashtags. *EECS, 545 final project, Fall 2011* (2011).

[35] Domingos, P. A unified bias-variance decomposition and its applications. In *ICML* (2000), Citeseer, pp. 231–238.

[36] Duin, R. P. W., and Tax, D. M. J. Classifier conditional posterior probabilities. Lecture Notes in Computer Science, Springer, pp. 611–619.

[37] Dwyer, R. A. Higher-dimensional voronoi diagrams in linear expected time. In *Proceedings of the fifth annual symposium on Computational geometry* (New York, USA, 1989), SCG '89, ACM, pp. 326–333.

[38] Dyer, R. *Self-Delaunay meshes for surfaces.* Simon Fraser University, Burnaby, BC, CANADA, 2010.

[39] Edelsbrunner, H., Kirkpatrick, D., and Seidel, R. On the shape of a set of points in the plane. *IEEE Trans. Inf. Theor.* (2006), 551–559.

[40] Efron, B., and Tibshirani, R. J. *An Introduction to the Bootstrap.* Chapman & Hall, New York, 1993.

[41] Eppstein, D., Paterson, M., and Yao, F. F. On nearest-neighbor graphs. *Discrete and Computational Geometry* (1997), 263–282.

[42] Esuli, A., and Sebastiani, F. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06* (2006), pp. 417–422.

[43] Farris, F. A. The gini index and measures of inequality. *The American Mathematical Monthly 117*, 10 (2010), 851–864.

[44] Ferrandiz, S., and Boull, M. Utilisation des graphes de proximit dans le cadre de l'apprentissage bas sur les voisins. In *EGC* (2004), vol. RNTI-E-2 of *Revue des Nouvelles Technologies de l'Information*, Cpadus-ditions, pp. 355–366.

[45] Fradkin, D., and Madigan, D. Experiments with random projections for machine learning. In *Proceedings of the 9th ACM SIGKDD inter. conf. on Knowledge discovery and data mining* (2003), ACM, p. 522.

[46] Freund, Y., and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences 55* (1997), 119 – 139.

[47] Freund, Y., and Schapire, R. E. A short introduction to boosting. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (1999), Morgan Kaufmann, pp. 1401–1406.

[48] Friedman, N., Geiger, D., Goldszmidt, M., Provan, G., Langley, P., and Smyth, P. Bayesian network classifiers. In *Machine Learning* (1997), pp. 131–163.

[49] Gabow, H. N., Bentley, J. L., and Tarjan, R. E. Scaling and related techniques for geometry problems. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1984), STOC '84, ACM, pp. 135–143.

[50] Gabriel, K. R., and Sokal, R. R. A new statistical approach to geographic variation analysis, 1969.

[51] Gabrilovich, E., and Markovitch, S. Wikipedia-based semantic interpretation for natural language processing. *J. Artif. Int. Res. 34*, 1 (Mar. 2009), 443–498.

[52] Gader, P. D., Mohamed, M. A., and Keller, J. M. Fusion of handwritten word classifiers. *Pattern Recognition Letters* (1996), 577–584.

[53] Gelman, A., Carlin, J., Stern, H., and Rubin, D. *Bayesian Data Analysis, Second Edition.* Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2003.

[54] Giudici, P., and Figini, S. *Applied Data Mining for Business and Industry.* Wiley, 2009.

[55] Glance, N., Hurst, M., Nigam, K., Siegler, M., Stockton, R., and Tomokiyo, T. Deriving marketing intelligence from online discussion. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (New York, NY, USA, 2005), KDD '05, ACM, pp. 419–428.

[56] Go, A., Bhayani, R., and Huang, L. Twitter sentiment classification using distant supervision. *Processing* (2009), 1–6.

[57] Hastie, T., and Tibshirani, R. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Anal. Mach. Intell.* (1996), 607–616.

[58] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning.* Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[59] Ho, T. Nearest neighbors in random subspaces. *Advances in Pattern Recognition* (1998), 640–648.

[60] Ho, T. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 8 (1998), 832–844.

[61] Ho, T., and Kleinberg, E. Building projectable classifiers of arbitrary complexity. In *International Conference on Pattern Recognition* (1996), vol. 13, pp. 880–885.

[62] Hsu, C.-W., and Lin, C.-J. A comparison of methods for multiclass support vector machines, 2002.

[63] Huang, C.-L., and jen Wang, C. A ga-based feature selection and parameters optimization for support vector machines, 2006.

[64] Huang, Y., and Suen, C. The behavior-knowledge space method for combination of multiple classifiers. In *Computer Vision and Pattern Recognition* (1993), IEEE Computer Society Conference, pp. 347–352.

[65] Huberman, B. A., Romero, D. M., and Wu, F. Social networks that matter: Twitter under the microscope. *CoRR abs/0812.1045* (2008).

[66] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural Comput.* (1991), 79–87.

[67] Jaromczyk, J. W., and Toussaint, G. T. Relative neighborhood graphs and their relatives. In *Proc. IEEE* (1992), pp. 1502–1517.

[68] Java, A., Song, X., Finin, T., and Tseng, B. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis* (New York, NY, USA, 2007), WebKDD/SNA-KDD '07, ACM, pp. 56–65.

[69] Jiang, L., Yu, M., Zhou, M., Liu, X., and Zhao, T. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1* (Stroudsburg, PA, USA, 2011), HLT '11, Association for Computational Linguistics, pp. 151–160.

[70] John, G., and Langley, P. Estimating continuous distributions in bayesian classifiers. In *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (1995), Morgan Kaufmann, pp. 338–345.

[71] Josh Weissbock, A. A. A. E., and Inkpen, D. Using external information for classifying tweets. BRACIS'2.

[72] Kas, G. V. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics, Vol. 29, No. 2. (1980), pp. 119-127.*.

[73] Katajainen, J., and Nevalainen, O. An almost naive algorithm for finding relative neighbourhood graphs in lp metrics. *ITA*, 2 (1987), 199–215.

[74] Kearns, M., and Mansour, Y. On the boosting ability of top-down decision tree learning algorithms. In *In Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing* (1995), ACM Press, pp. 459–468.

[75] Kearns, M., and Valiant, L. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM* (1994), 67–95.

[76] Kim, D., Jo, Y., Moon, I.-C., and Oh, A. Analysis of twitter lists as a potential source for discovering latent characteristics of users. In *Workshop on Microblogging at the ACM Conference on Human Factors in Computer Systems. (CHI 2010)* (2010).

[77] Kim, H.-C., and Ghahramani, Z. Bayesian classifier combination. *Journal of Machine Learning Research - Proceedings Track* (2012), 619–627.

[78] Kinsella, S., Passant, A., and Breslin, J. G. Topic classification in social media using metadata from hyperlinked objects. In *Proceedings of the 33rd European conference on Advances in information retrieval* (Berlin, Heidelberg, 2011), ECIR'11, Springer-Verlag, pp. 201–206.

[79] Kirkpatrick, D., and Radke, J. *A Framework for Computational Morphology.* North-Holland, 1984.

[80] Kittler, J., Hatef, M., Duin, R. P. W., and Matas, J. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.* (1998), 226–239.

[81] Kleinberg, E. An overtraining-resistant stochastic modeling method for pattern recognition. *The annals of statistics 24*, 6 (1996), 2319–2349.

[82] Kleinberg, E. On the algorithmic implementation of stochastic discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 5 (2000), 473–490.

[83] Kohavi, R., and Wolpert, D. Bias plus variance decomposition for zero-one loss functions. In *Machine Learning-International Workshop* (1996), Citeseer, pp. 275–283.

[84] Korting., T. S. C4.5 algorithm and multivariate decision trees image processing division. *National Institute for Space Research INPE*.

[85] Krishnamurthy, B., Gill, P., and Arlitt, M. A few chirps about twitter. In *Proceedings of the first workshop on Online social networks* (New York, NY, USA, 2008), WOSN '08, ACM, pp. 19–24.

[86] Kuncheva, L., Kounchev, R., and Zlatev, R. Aggregation of multiple classification decisions by fuzzy templates. *Third European Congress on Intelligent Technologies and Soft Computing EUFIT'95* (1995).

[87] Kuncheva, L. I. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion* (2002), 135–148.

[88] Kuncheva, L. I. *Combining Pattern Classifiers: Methods and Algorithms.* Wiley-Interscience, 2004.

[89] Kuncheva, L. I., Bezdek, J. C., and Duin, R. P. W. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition 34* (2001), 299–314.

[90] Kwak, H., Lee, C., Park, H., and Moon, S. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web* (New York, NY, USA, 2010), WWW '10, ACM, pp. 591–600.

[91] Lam, L. Classifier combinations: Implementations and theoretical issues. In *Proceedings of the First International Workshop on Multiple Classifier Systems* (London, UK, 2000), MCS '00, Springer-Verlag, pp. 77–86.

[92] Lam, L., and Suen, C. Optimal combination of pattern classifiers. *Pattern Recognition Letters* (1995), 945–954.

[93] Latourrette, M. Toward an ecplanatory similarity measure for nearest-neighbor classification. In *ECML* (2000), pp. 238–245.

[94] Lee, K., Palsetia, D., Narayanan, R., Patwary, M. M. A., Agrawal, A., and Choudhary, A. Twitter trending topic classification. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops* (Washington, DC, USA, 2011), ICDMW '11, IEEE Computer Society, pp. 251–258.

[95] Leplumey, I., and Queguiner, C. Un graphe de voisinage basé sur l'utilisation des distances discrètes. In *CIFED 2002 - Colloque international francophone sur l'ecrit et le document* (Hammamet, Tunisia, 2002).

[96] Lévy, B., and Liu, Y. Lp centroidal voronoi tessellation and its applications. *ACM Trans. Graph.* (2010), 119:1–119:11.

[97] Liaw, A., and Wiener, M. Classification and regression by randomforest. *R news 2*, 3 (2002), 18–22.

[98] Lim, K. H., and Datta, A. Interest classification of twitter users using wikipedia. In *Proceedings of the 9th International Symposium on Open Collaboration* (New York, NY, USA, 2013), WikiSym '13, ACM, pp. 22:1–22:2.

[99] Lffler, M., and Snoeyink, J. Delaunay triangulation of imprecise points in linear time after preprocessing. *Comput. Geom.* (2010), 234–242.

[100] Maimon, O., and Rokach, L. *Data Mining and Knowledge Discovery Handbook*. The Kluwer International Series in Engineering and Computer Science. Springer, 2005.

[101] Maria, C. Représentation compacte de la triangulation de delaunay. *Topological and Geometric Data Analysis* (2009).

[102] Matula, D. W., and Sokal, R. R. Properties of gabriel graphs relevant to geographic variation research and the clustering of points in the plane. *Geographical Analysis* (1980), 205–222.

[103] Maynard, D., and Funk, A. Automatic detection of political opinions in tweets. In *Proceedings of the 8th international conference on The Semantic Web* (Berlin, Heidelberg, 2012), ESWC'11, Springer-Verlag, pp. 88–99.

[104] Michalke, M. *koRpus: An R Package for Text Analysis*, 2013. (Version 0.04-40).

[105] Michelson, M., and Macskassy, S. A. Discovering users' topics of interest on twitter: a first look. In *Proceedings of the fourth workshop on Analytics for noisy unstructured text data* (New York, NY, USA, 2010), AND '10, ACM, pp. 73–80.

[106] Millman, D. L., and Verma, V. A slow algorithm for computing the gabriel graph with double precision. In *CCCG* (2011).

[107] Muhlenbach, F. Méthode de regroupement par graphe de voisinage. In *EGC* (2009), Revue des Nouvelles Technologies de l'Information, Cépaduès-Editions, pp. 361–366.

[108] Muhlenbach, F., and Rakotomalala, R. Multivariate supervized discretization, a neighborhood graph approach. In *3rd International Conference on Data mining (ICDM'02)* (2002), IEEE, pp. 314–321.

[109] Nguyen, C. H., and Ho, T. B. An efficient kernel matrix evaluation measure. *Pattern Recogn.* (2008), 3366–3372.

[110] of Defense, U. S. D. *Sampling procedures and tables for inspection by attributes.* Military standard. Dept. of Defense, 1989.

[111] Olaru, C., and Wehenkel, L. A complete fuzzy decision tree technique, 2003.

[112] Opitz, D., and Maclin, R. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research 11*, 169-198 (1999), 12.

[113] Pang, B., and Lee, L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *In Proceedings of the ACL* (2004), pp. 271–278.

[114] Pang, B., and Lee, L. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr. 2*, 1-2 (Jan. 2008), 1–135.

[115] Pang, B., Lee, L., and Vaithyanathan, S. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP* (2002), pp. 79–86.

[116] Pennacchiotti, M., and Popescu, A.-M. Democrats, republicans and starbucks afficionados: user classification in twitter. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2011), KDD '11, ACM, pp. 430–438.

[117] Pettie, S., and Ramachandran, V. An optimal minimum spanning tree algorithm. *J. ACM* (2002), 16–34.

[118] Phan, X.-H., Nguyen, L.-M., and Horiguchi, S. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web* (New York, NY, USA, 2008), WWW '08, ACM, pp. 91–100.

[119] Phelan, O., McCarthy, K., and Smyth, B. Using twitter to recommend real-time topical news. In *Proceedings of the third ACM conference on Recommender systems* (New York, NY, USA, 2009), RecSys '09, ACM, pp. 385–388.

[120] Pisetta, V., and Zighed, D. A. Similarity and kernel matrix evaluation based on spatial autocorrelation analysis. In *ISMIS* (2009), J. Rauch, Z. W. Ras, P. Berka, and T. Elomaa, Eds., Lecture Notes in Computer Science, Springer, pp. 422–430.

[121] Preparata, F., and Shamos, M. *Computational geometry: an introduction*. Springer, 1985.

[122] Qi, X., and Davison, B. D. Classifiers without borders: incorporating fielded text from neighboring web pages. In *SIGIR* (2008), S.-H. Myaeng, D. W. Oard, F. Sebastiani, T.-S. Chua, and M.-K. Leong, Eds., ACM, pp. 643–650.

[123] Quinlan, J. R. Induction of decision trees. *Mach. Learn.* (1986), 81–106.

[124] Quinlan, J. R. Simplifying decision trees, 1986.

[125] Quinlan, J. R. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA, 1993.

[126] Quinlan, J. R., and Rivest, R. L. Inferring decision trees using the minimum description length principle. Technical memo, Massachusetts Institute of Technology, Laboratory for Computer Science, 1987.

[127] Read, J. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL Student Research Workshop* (Stroudsburg, PA, USA, 2005), ACLstudent '05, Association for Computational Linguistics, pp. 43–48.

[128] Ritter, G., Woodruff, H., Lowry, S., and Isenhour, T. An algorithm for a selective nearest neighbor decision rule (corresp.). *Information Theory, IEEE Transactions on* (1975), 665–669.

[129] Rokach, L. *Data Mining with Decision Trees: Theory and Applications*. Series in machine perception and artificial intelligence. World Scientific Publishing Company, Incorporated, 2008.

[130] Schapire, R. The strength of weak learnability. *Machine Learning* (1990), 197–227.

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

[132] Shawe-Taylor, J., and Cristianini, N. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, USA, 2004.

[133] Shewchuk, J. R. Lecture notes on delaunay mesh generation. Tech. rep., 1999.

[134] Shipp, C., and Kuncheva, L. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion 3*, 2 (2002), 135–148.

[135] Simpson, E., Roberts, S. J., Smith, A., and Lintott, C. Bayesian combination of multiple, imperfect classifiers. In *NIPS 2011* (Oxford, 2011), University of Oxford, pp. 1–8.

[136] Skurichina, M., and Duin, R. Bagging, boosting and the random subspace method for linear classifiers. *Pattern Analysis & Applications 5*, 2 (2002), 121–135.

[137] Song, Y., Huang, J., Zhou, D., Zha, H., and Giles, C. L. Iknn: Informative k-nearest neighbor pattern classification. In *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases* (Berlin, Heidelberg, 2007), PKDD, Springer-Verlag, pp. 248–264.

[138] Staff, B. S. I. *Sequential Sampling Plans for Inspection by Variables for Percent Nonconforming (known Standard Deviation)*. B S I Standards, 2008.

[139] Su, T.-H., and Chang, R.-C. On constructing the relative neighborhood graphs in euclideank-dimensional spaces. *Computing* (1991), 121–130.

[140] Takuya Yamashita, Haruhiko Sato, S. O., and Kurihara, M. Classication of twitter users based on following relations. *Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I, IMECS 2013 Hong Kong* (2013).

[141] Tan, C., Lee, L., Tang, J., Jiang, L., Zhou, M., and Li, P. User-level sentiment analysis incorporating social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2011), KDD '11, ACM, pp. 1397–1405.

[142] Toussaint, G. Comment: Algorithms for computing relative neighbourhood graph. *Electronics Letters* (1980), 860–860.

[143] Toussaint, G. Proximity graphs for nearest neighbor decision rules: Recent progress. In *Progress, Proceedings of the 34 th Symposium on the INTERFACE* (2002), pp. 17–20.

[144] Toussaint, G., Bhattacharya, B., and Poulsen, R. The application of voronoi diagrams to non-parametric decision rules. *Proceedings of the 16th Symposium on Computer Science and Statistics* (1984).

[145] Toussaint, G., and Foulsen, R. Some new algorithms and software implementation methods for pattern recognition research. In *Computer Software and Applications Conference, 1979. Proceedings. COMPSAC 79. The IEEE Computer Society's Third International* (1979), pp. 55–58.

[146] Toussaint, G. T. Pattern Recognition and Geometrical Complexity. *NATO Conference Series, (Series) 4: Marine Sciences* (1980), 1324–1347.

[147] Toussaint, G. T. The relative neighbourhood graph of a finite planar set. *Pattern Recognition* (1980), 261–268.

[148] Toussaint, G. T. A graph-theoretical primal sketch. *G.T. Toussaint, editor, Computational Morphology* (1988), 229–260.

[149] Toussaint, G. T. Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining. *Int. J. Comput. Geometry Appl.* (2005), 101–150.

[150] Toussaint, G. T., and Menard, R. Fast algorithms for computing the planar relative neighborhood graph. *Methods of Operations Research.* (1981), 425–428.

[151] Tumasjan, A., Sprenger, T., Sandner, P., and Welpe, I. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media* (2010), pp. 178–185.

[152] Vessereau, A. Une collection de plans d'echantillonnage pour les controles de reception. *Revue de Statistique Appliquee 36*, 2 (1988), 5–22.

[153] Watson, D. F. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The Computer Journal* (1981), 167–172.

[154] Weinberger, K. Q., and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.* (2009), 207–244.

[155] Wiebe, J., and Riloff, E. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of the 6th international conference on Computational Linguistics and Intelligent Text Processing* (Berlin, Heidelberg, 2005), CICLing'05, Springer-Verlag, pp. 486–497.

[156] Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics Bulletin* (1945), 80–83.

[157] Wilson, D. L. Asymptotic properties of nearest neighbor rules using edited data. *Systems, Man and Cybernetics, IEEE Transactions on* (1972), 408–421.

[158] Wilson, T., Wiebe, J., and Hoffmann, P. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing* (Stroudsburg, PA, USA, 2005), HLT '05, Association for Computational Linguistics, pp. 347–354.

[159] Woods, K. S., Kegelmeyer, W. P., and Bowyer, K. W. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. Pattern Anal. Mach. Intell.* (1997), 405–410.

[160] Xu, L., Krzyzak, A., and Suen, C. Y. Methods of combining multiple classifiers and their applications to handwriting recognition. In *IEEE Transactions on Systems, Man, and Cybernetics* (1992), pp. 418–435.

[161] Yu, B., Kaufmann, S., and Diermeier, D. Exploring the characteristics of opinion expressions for political opinion classification. In *Proceedings of the 2008 international conference on Digital government research* (2008), dg.o '08, Digital Government Society of North America, pp. 82–91.

[162] Zahn, C. Graph-theoretical methods for detecting and describing gestalt clusters. *Computers, IEEE Transactions on* (1971), 68–86.

[163] Zhang, H. The optimality of naive bayes. In *FLAIRS Conference* (2004), pp. 562–567.

[164] Zhou, Z., Wu, J., and Tang, W. Ensembling neural networks: Many could be better than all* 1. *Artificial intelligence 137*, 1-2 (2002), 239–263.

[165] Zhou, Z.-H. Ensemble learning. In *Encyclopedia of Biometrics*. Springer US, 2009.

[166] Zhou, Z.-H. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Serie. Taylor & Francis, 2012.

[167] Zighed, D., Ezzeddine, D., and Rico, F. Neighborhood random classification. In *Advances in Knowledge Discovery and Data Mining*, P.-N. Tan, S. Chawla, C. Ho, and J. Bailey, Eds., vol. 7301 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 98–108.

[168] Zighed, D. A., Lallich, S., and Muhlenbach, F. Separability index in supervised learning. In *PKDD* (2002), T. Elomaa, H. Mannila, and H. Toivonen, Eds., Lecture Notes in Computer Science, Springer, pp. 475–487.

[169] Zighed, D.-A., Lallich, S., and Muhlenbach, F. A statistical approach of class separability. *Applied Stochastic Models in Business and Industry* (2005), 187–197.

Résumé

L'Apprentissage Supervisé est un domaine populaire de l'Apprentissage Automatique en progrès constant depuis plusieurs années. De nombreuses techniques ont été développées pour résoudre le problème de classification, mais, dans la plupart des cas, ces méthodes se basent sur la présence et le nombre de points d'une classe donnée dans des zones de l'espace que doit définir le classifieur. À cause de cela la construction de ce classifieur est dépendante de la densité du nuage de points des données de départ.

Dans cette thèse, nous montrons qu'utiliser la topologie des données peut être une bonne alternative lors de la construction des classifieurs. Pour cela, nous proposons d'utiliser les graphes topologiques comme le Graphe de Gabriel (GG) ou le Graphes des Voisins Relatifs (RNG). Ces dernier représentent la topologie de données car ils sont basées sur la notion de voisinages et ne sont pas dépendant de la densité. Pour appliquer ce concept, nous créons une nouvelle méthode appelée Classification aléatoire par Voisinages (**Random Neighborhood Classification (RNC)**). Cette méthode utilise des graphes topologiques pour construire des classifieurs. De plus, comme une Méthodes Ensemble (EM), elle utilise plusieurs classifieurs pour extraire toutes les informations pertinentes des données. les EM sont bien connues dans l'Apprentissage Automatique. Elles génèrent de nombreux classifieurs à partir des données, puis agrègent ces classifieurs en un seul. Le classifieur global obtenu est reconnu pour être très efficace, ce qui a été montré dans de nombreuses études. Cela est possible car il s'appuie sur des informations obtenues auprès de chaque classifieur qui le compose.

Nous avons comparé RNC à d'autres méthodes de classification supervisées connues sur des données issues du référentiel *UCI* Irvine. Nous constatons que RNC fonctionne bien par rapport aux meilleurs d'entre elles, telles que les Forêts Aléatoires (RF) et Support Vector Machines (SVM). La plupart du temps, RNC se classe parmi les trois premières méthodes en terme d'efficacité. Ce résultat nous a encouragé à étudier RNC sur des données réelles comme les tweets.

Twitter est un réseau social de micro-blogging. Il est particulièrement utile pour étudier l'opinion à propos de l'actualité et sur tout sujet, en particulier la politique. Cependant, l'extraction de l'opinion politique depuis Twitter pose des défis particuliers. En effet, la taille des messages, le niveau de langage utilisé et ambiguïté des messages rend très difficile d'utiliser les outils classiques d'analyse de texte basés sur des calculs de fréquence de mots ou des analyses en profondeur de phrases. C'est cela qui a motivé cette étude. Nous proposons d'étudier les **couples** auteur/sujet pour classer le tweet en fonction de l'opinion de son auteur à propos d'un politicien (un sujet du tweet). Nous

proposons une procédure qui porte sur l'identification de ces opinions. Nous pensons que les tweets expriment rarement une opinion objective sur telle ou telle action d'un homme politique mais plus souvent une conviction profonde de son auteur à propos d'un mouvement politique. Détecter l'opinion de quelques auteurs nous permet ensuite d'utiliser la similitude dans les termes employés par les autres pour retrouver ces convictions à plus grande échelle. Cette procédure à 2 étapes, tout d'abord identifier l'opinion de quelques couples de manière semi automatique afin de constituer un référentiel, puis ensuite d'utiliser l'ensemble des tweets d'un couple (tous les tweets d'un auteur mentionnant un politicien) pour les comparer avec ceux du référentiel.

L'Apprentissage Topologique semble être un domaine très intéressant à étudier, en particulier pour résoudre les problèmes de classification. Plusieurs concept pour obtenir des informations à partir de graphes topologiques restent à analyser comme celles décrites par Aupetit, M. dans son travail (2005), mais notre travail montrent que l'Apprentissage Topologique peut être un moyen efficace pour résoudre le problème de classification.

Mots-Clés: Apprentissage Automatique, Classification, Graphes de voisinages, Graphe de Gabriel, Graphe des voisins relatifs, Méthodes d'ensemble, Naive Bayes, Décision Templates, Twitter, Opinion Mining.