Possibilistic Classifiers for Certain/Uncertain Numerical Data

Myriam Bounhas

Université de Tunis
Institut Supérieur de Gestion
Ecole Doctorale Sciences de Gestion

Possibilistic Classifiers for Certain/ Uncertain Numerical Data

THESE
En vue de l'obtention du Doctorat
En Informatique de Gestion

Présentée et soutenue publiquement par:
Myriam BOUNHAS ELAYEB

Le 10 janvier 2013

Membres du Jury:

Année Universitaire 2012 / 2013

• *M. Abdelwahed TRABELSI*	*:Professeur*	*à l'Université de Tunis*	*:Président*
• *M. Khaled MELLOULI*	*:Professeur*	*à l'Université de Carthage*	*:Directeur de thèse*
• *M. Henri PRADE*	*:Dir. CNRS*	*à l'IRIT Toulouse*	*: Co-directeur*
• *M. Mathieu SERRURIER*	*:M.de conf*	*à l'UPS Toulouse*	*: Co-directeur*
• *Mme Nahla BEN AMOR*	*:Professeur*	*à l'Université de Tunis*	*:Rapporteur*
• *Mme Sandra SANDRI*	*:M.de conf*	*à LAC/INPE, Brésil*	*:Rapporteur*
• *M. Zied ELOUEDI*	*:Professeur*	*à l'Université de Tunis*	*:Membre*

In memory of my father,

To my mother, for all her sacrifices and patience,

To my husband and my daughters Israa and Yasmine for being great pillars of support and love,

To all my brothers and sisters for their encouragement throughout the course of this thesis,

To my lovely Siwar for her unconditional love.

Myriam Bounhas

December 7, 2012

Acknowledgements

Most of all thanks to God who allowed me to enter into the path of my fulfillment after several years. I owe my gratitude to all those people who have contributed to the production of this dissertation and because of whom my graduate experience has been developed.

Foremost, I would like to thank my advisor Pr. Khaled Mellouli. I have been fortunate to have an advisor who gave me the freedom to explore on my own and at the same time the guidance to recover when my steps faltered. His patience and support helped me overcome many crisis situations especially at the beginning of this thesis.

My deepest gratitude is to my co-advisor Dr. Henri Prade for providing me with the opportunity to complete my PhD thesis as an external student at the IRIT Toulouse France. I especially want to thank my advisor whose support and guidance made my thesis work possible. He has been actively interested in my work and has always been available to advise me even it sometimes represents an imposition on his time to perform this. I am very grateful for his patience, motivation, enthusiasm, and immense experience that, taken together, make him a great mentor. Henri taught me how to express ideas and how strengthen my steps in research.

I owe a debt of gratitude to my co-advisor Dr. Mathieu Serrerier, that has been always there to listen and give advice. I am deeply grateful to him for his encouragement and for reading my reports; commenting on my views and helping me understand and enrich my ideas. I am also thankful to him for the long discussions that helped me sort out the technical details of my work.

Besides my advisors, I would like to thank the rest of my thesis committee: Pr. Abdelwahed Trabelsi, Pr. Nahla Ben Amor, Dr. Sandra Sandri, and Pr. Zied Elouedi for their encouragement, insightful comments, and interesting questions.

I wish to specially thank Dr. Didier Dubois, Dr. Eyke Hüllermeier, and Dr. Olivier Strauss, for numerous discussions and lectures on related topics that helped me improve my knowledge in the area and overcome some difficulties in my work.

My sincere thanks also go to Dr. Ibrahim Bounhas for his technical help and to Florence Boué for helping me in my research during my stages in IRIT Toulouse, France.

I want to express my thinks and love to my mother, my brothers and my aunt Tibra whose sacrifices and great efforts have enabled me to reach the present position in life. I am forever indebted to my mother for her unceasing prayers and patience.

There are no words to say thank you to my husband Dr. Bilel Elayeb who has been a true and great supporter during my good and bad times and has unconditionally helped me scientifically and financially to attain my goal. These past several years have not been an easy ride, both academically and personally. I truly thank Bilel for his sacrifices and for sticking by my side, even when I was irritable and depressed. This work would not have been possible without your support.

My sincere thanks also go to my sister in law Samiha that has took care to my daughter Israa during my studies, without her help this work would be more difficult. Finally, I would like to thank the family Elayeb, especially Ms. Fatma Ben Smida and Mr. Nafti Elayeb for their continuous encouragement.

Abstract

This thesis enters within the framework of Machine learning and concerns the study of a variety of *classification* methods for *numerical* data.

The first issue in this work concerns the study of Rule based classification. In fact, rule induction algorithms have gained a high popularity among machine learning techniques due to the "intelligibility" of their output, when compared to other "black-box" classification methods. However, they suffer from two main drawbacks when classifying test examples: i) the multiple classification problems when many rules cover an example and are associated with different classes, and ii) the choice of a default class, which concerns the non-covering case. Our fist contribution is to propose a family of *Possibilistic Rule-based Classifiers* (PRCs) to deal with such problems which are an extension and a modification of the PART algorithm. The PRCs keep the same rule learning step as PART, but differ in other respects. In particular, the PRCs learn fuzzy rules instead of crisp rules, consider weighted rules at deduction time in an unordered manner instead of rule lists and reduce the number of examples not covered by any rule using a fuzzy rule set with large supports. The experiments reported show that the PRCs lead to improve the accuracy of the classical PART algorithm.

On the other hand Naive Bayesian Classifiers (NBC), which relies on independence hypotheses, together with a normality assumption to estimate densities for numerical data, are known for their simplicity and their effectiveness. However estimating densities, even under the normality assumption, may be problematic in case of poor data. In such a situation, possibility distributions may provide a more faithful representation of these data. *Naive Possibilistic Classifiers* (NPC), based on possibility theory, have been recently proposed as a counterpart of Bayesian classifiers to deal with classification tasks. There are only few works that treat possibilistic classification and most of existing NPC deal only with categorical attributes.

A second contribution in this thesis focuses on the estimation of possibility distributions for continuous data. For this purpose we investigate two families of possibilistic classifiers. The first one is derived from classical or flexible Bayesian classifiers

by applying a probability-possibility transformation to Gaussian distributions which introduces some further tolerance in the description of classes and gives place to the *Naive Possibilistic Classifier* (NPC) and the *Flexible Naive Possibilistic Classifier* (FNPC). In the same context, we also use a probability-possibility transformation method enabling us to derive a possibilistic distribution as a family of Gaussian distributions. First, we have applied the transformation method to move from a classical NBC to NPC-2, which takes into account the confidence intervals of the Gaussian distributions. Then, we have tested the feasibility of a Flexible Naive Possibilistic Classifier (FNPC-2).

The second family of possibilistic classifiers abandons the normality assumption and has a direct representation of data. We propose two other classifiers named *Fuzzy Histogram Classifier* (FuHC) and *Nearest Neighbor-based Possibilistic Classifier* (NNPC) in this context. The two proposed classifiers exploit an idea of proximity between attribute values in order to estimate possibility distributions. We show that Possibilistic Classifiers have a better capability to detect new instances for which the classification is ambiguous than Bayesian classifiers, where probabilities may be poorly estimated and illusorily precise. Moreover, we propose, in this case, an hybrid Possibilistic Classification approach based on a *Nearest Neighbour Heuristics* to improve the accuracy of the proposed possibilistic classifiers when the available information is insufficient to choose between classes.

The last issue in this thesis concerns the classification of data with continuous input variables in presence of uncertainty. In many real-world problems, input data may be pervaded with uncertainty. Naive possibilistic classifiers have been proposed as a counterpart to Bayesian classifiers to deal with classification tasks in presence of uncertainty for categorical data. Following this line, we extend possibilistic classifiers that we have previously proposed for numerical data, in order to cope with uncertainty in data representation. We consider two types of uncertainty: i) the uncertainty associated with the class in the training set, which is modelled by a possibility distribution over class labels, and ii) the imprecision pervading attribute values in the testing set represented under the form of intervals for continuous data. We first adapt the possibilistic classification model, previously proposed for the certain case, in order to accommodate the uncertainty about class labels. Then, we propose an extension principle-based algorithm to deal with imprecise attribute values.

Possibilistic classifiers are compared to classical or flexible Bayesian classifiers on a collection of benchmarks databases. The experiments reported show the interest of possibilistic classifiers. In particular, flexible possibilistic classifiers perform well for data agreeing with the normality assumption, while proximity-based possibilistic classifiers outperform others in the other cases. On the other hand, results in the

uncertain case show the efficiency of possibilistic classifiers for handling uncertainty in data. In particular, the probability-to-possibility transform-based classifiers show a robust behaviour when dealing with imperfect data.

Key words: Naive Possibilistic Classifier, Possibility Theory, Gaussian distribution, Proximity, Naive Bayesian Classifier, Numerical Data, Uncertainty, Possibilistic Rule-based Classifier, Fuzzy Rules, Decision List.

Résumé

Le sujet de cette thèse entre dans le cadre d'apprentissage automatique et concerne l'étude d'une variété de méthodes de classification pour les données numériques.

Notre premier intérêt dans ce travail concerne l'étude des algorithmes d'induction des règles et leurs performances lors de la classification. Ces algorithmes gagnent plus de popularité parmi les autres méthodes d'apprentissage automatique grâce à leurs capacités d'interprétation. Toutefois, l'étude de ces classifieurs nous a montré qu'ils souffrent principalement de deux limites : (i) le problème de classification multiple qui se produit quand plusieurs règles couvrent l'exemple à classer mais qui ont différent conséquents (classes) et (ii) le problème de non couverture quand aucune règle ne couvre l'exemple et qui concerne le choix de la classe/règle par défaut. Notre contribution dans ce cadre consiste à proposer une famille de Classifieur à base de Règles Possibiliste (Possibilistic Rule-based Classifiers (PRCs)) pour traiter ce type de problèmes qui est une extension et une modification de l'algorithme PART. Le classifieur PRC garde le même principe d'apprentissage que celui de PART et diffère de ce dernier en plusieurs aspects. En particulier, le PRC apprend des règles floues au lieu des règles classiques, considère des règles non ordonnées assignées à des poids au moment de la déduction au lieu de listes de décision et réduit le nombre d'exemples non couverts par aucune règle en utilisant un ensemble de règles floues à large support. Les expérimentations ont montré l'efficacité des PRCs d'améliorer le taux de classification si comparé à l'algorithme PART classique.

D'autres parts, les classifieurs Bayésiens Naïfs (NBC) ont été largement utilisés dans plusieurs domaines pour classer les données numériques. Ces classifieurs se basent sur l'hypothèse d'indépendance et l'hypothèse de normalité pour estimer les densités des probabilités des attributs. En fait, ces deux hypothèses sont limitantes dans le sens qu'elles peuvent être problématique dans le cas où les données sont très réduites. Dans ce genre de cas, les distributions de possibilités peuvent offrir une meilleure représentation de ces données.

Les Classifieurs Possibilistes Naïfs (NPC), basés sur la théorie de possibilité, ont été récemment proposés comme une contre partie des classifieurs Bayésiens pour

traiter les problèmes de classification. Il existe uniquement très peu de travaux qui s'intéressent à la classification possibiliste dans la littérature malgré que la théorie des possibilités est connue comme un outil convenable pour le traitement des données réduites et/ou imparfaites. Deux sous-problèmes sont principalement traités dans ce cadre : (i) l'estimation d'une distribution de possibilité associée aux attributs numériques qui doit être la plus représentative des données réelles et (ii) la modélisation et le traitement d'incertitude reliée aux attributs et à la classe lors de l'estimation de cette distribution.

Une deuxième contribution dans cette thèse consiste à estimer les distributions de possibilités pour les données continues. Dans ce cadre, nous avons proposé et étudié deux familles des classifieurs possibilistes pour des attributs numériques. La première famille, adoptant l'hypothèse de normalité des distributions des données, est basée sur une méthode de transformation de probabilité en possibilité permettant de transformer un Classifieur Bayésien Naïf classique en un Classifieur Possibiliste Naïf (Naïf Possibilistic Classifier: NPC). Cette transformation a l'avantage d'ajouter plus de tolérance dans la description des classes. Nous avons également analysé la faisabilité d'un Classifieur Possibiliste Naïf Flexible (Flexible Naïf Possibilistic Classifier : FNPC) qui constitue la contre partie possibiliste du Classifieur Bayésien Naïf Flexible. Dans le même contexte, nous avons aussi utilisé une deuxième méthode de transformation de probabilité en possibilité permettant de dériver une distribution de possibilité comme une famille de distributions Gaussiennes. En premier lieu, nous avons transformé le classifieur NBC classique en un NPC-2, qui prend en considération l'intervalle de confiance des distributions Gaussiennes. Ensuite nous avons testé la faisabilité d'un Classifieur Possibiliste Naïf Flexible (FNPC-2).

La deuxième famille de classifieurs possibilistes abandonne l'hypothèse de normalité et reflète une représentation directe et plus proche sur les données. Nous avons proposé deux classifieurs appelés Classifieur à base des Histogrammes Floues (Fuzzy Histogram Classifier : FuHC) et Classifieur Possibiliste à base du plus Proche Voisin (Nearest Neighbor-based Possibilistic Classifier: NNPC). Le premier classifieur exploite une idée de proximité entre les valeurs d'attribut d'une façon additive tandis que le second est basé seulement sur l'analyse des proximités entre les attributs sans les compter.

Nous avons montré que les classifieurs possibilistes possèdent une meilleure capacité de détecter l'ambiguïté lors la classification que celui des classifieurs Bayésiens, pour lesquels les probabilités, estimées à partir des données limitées, sont illusoirement précises. Dans le but d'améliorer la performance des classifieurs possibilistes, nous avons proposé une approche de classification possibiliste hybride basée sur une heuristique à base de plus proche voisin. Cette heuristique permet d'augmenter l'exactitude de ces classifieurs dans le cas où le classifieur possibiliste est incapable

de distinguer entre deux classes à plausibilités proches.

Une troisième contribution principale dans cette thèse est le traitement d'incertitude lors de la classification des données numériques utilisant la théorie de possibilité. Dans ce cadre, nous avons étendu les classifieurs possibilistes proposés dans le cas certain pour être compatibles en présence d'incertitude aux niveaux des données. Nous avons considéré deux types d'incertitude : (i) l'incertitude reliée à la classe dans l'ensemble d'apprentissage modélisée à travers une distribution de possibilité sur les valeurs de la classe et (ii) l'imprécision au niveau des valeurs d'attributs dans l'ensemble de test représentée sous forme d'intervalles dans le cas continu. En premier lieu, nous avons adapté le modèle de classification possibiliste, précédemment développé pour le cas certain, pour prendre en considération l'incertitude au niveau des valeurs de la classe. En second lieu, nous avons aussi proposé un algorithme basé sur le principe d'extension qui consiste à estimer les distributions des possibilités pour un attribut incertain (intervalle) en regardant les distributions des possibilités de chaque attribut dans l'ensemble d'apprentissage appartenant à cet intervalle.

Nous avons implémenté et testé les différentes familles de classifieurs possibilistes sur une collection de bases de données. Une étude comparative entre les classifieurs possibilistes proposés et les approches de classification Bayésiènnes classiques a montré l'efficacité des classifieurs possibilistes pour traiter les données continues. En particulier, les Classifieurs Possibilistes Flexibles sont performants dans le cas des données vérifiant l'hypothèse de normalité, tandis que les classifieurs possibiliste à base de proximité sont meilleurs dans les autres cas. D'autres parts, les résultats dans le cas incertain ont montré l'efficacité des classifieurs possibiliste lors du traitement d'incertitude au niveau des données. En particulier, les classifieurs basés sur la transformation de probabilité en possibilité sont robustes lors de la classification des données imparfaites.

Key words: Classifieur Possibiliste Naïf, Théorie des Possibilités, Distribution gaussiènne, Proximité, Classifieur Bayésien Naïf, Données numériques , Incertitude, Classifieur à base de Règles Possibiliste, Règles floues, Listes de décision.

Table of Contents

List of Figures

List of Tables

Introduction

Panoply of artificial intelligence techniques has been developed in order to imitate intelligent human behaviour. Using the available knowledge on a given problem, these techniques try to reproduce the process of human reasoning as close as possible. The general issue of simulating intelligence has been applied in a number of specific sub-problems such as: deduction, learning, reasoning, decision making, and knowledge representation.

There is no established unifying theory that guides AI research. At its beginning, research field in AI claims that human intelligence could be reduced to symbol exploration using *symbolic* methods. However, the development of symbolic systems has showed that they would never be able to imitate all processes of human cognition, especially pattern recognition, learning, and robotics. Many researchers are rather interested to $sub - symbolic$ methods such as connexionist methods and also *statistical* methods (probabilistic methods). Each of these approaches has its advantages, but also some drawbacks. Recently, there has been extensive research activity at combining (or integrating) symbolic and statistical (or connexionist) approaches. What they do is a kind of mapping from symbolic rules to other formalisms in order to gain from the complementarily between these methods and so help to improve their ability to reproduce human reasoning. From the beginning, *machine learning* has been central to AI research. Machine learning consists in the development of programs which can be improved by experience. The applications are numerous and concern a variety of fields. For example, we can cite the pattern recognition, especially text and voice recognition, the process control and the diagnosis of breakdowns, game programs.

Learning methods from examples are very used in information retrieval applied to a big mass of data. Indeed, the evolution of data processing nowadays enables to handle a data with a very big size such as $DataWarehouse$. For example, supermarkets can memorize large quantities of data concerning consumers and their purchases. The development of Internet and Intranet technologies makes that a heterogeneous variety of data resulting from different sources and in varied formats become accessible. The process of information retrieval in large quantity of data (KDD: Knowledge Discovery in Databases) includes various steps: data selection

(extraction of information from the Data warehouse); data preparation (removal of redundancy and noises in data), the data coding (normalization, choice of coding,...); the data extraction known as: $DataMining$.

This extraction phase exploits common tools for interrogation such as standard SQL query and the multidimensional queries, but also, for the extraction of *hidden* information, the learning algorithms are very useful. There are different domain of interest in machine learning, especially supervised, unsupervised and semi-supervised learning depending on whether outputs (for the given problem) are given or not for the learner when accomplishing the learning task. Among these domains we are interested to supervised learning and especially to $Classification$.

The purpose of classification methods is to identify the class to which belongs an object given certain descriptive features. They are applied to a large number of human activities and are in particular appropriated to the problem of automatic decision making. For example, one can establish a medical diagnosis from clinical description of a patient; answer to the request of a customer for a bank loan according to his personal situation, start an alarm process according to signals received by sensors. A first possible solution to solve this type of problem is *expert systems* based approaches. Within this framework, the knowledge of an expert (or a group of experts) is described through rules.

This set of rules forms an expert system which is used to classify new cases. This approach, largely used in the Eighties years, strongly depends on the capacity to extract and formalize knowledge from the expert. Other approaches are largely considered thereafter for which the procedure of classification will be automatically extracted from a set of examples. An example contains a set of descriptive features with the corresponding target (class).

For example, given the history of the loans accorded to each customer with the personal information for each customer and also the result of the loan request. Then, a learning system starts from this set of examples can execute a classification process and will be able to decide the attribution or not of the loan to a new customer according to its personal information. The main issue of classification is to induce a general procedure to classify a new example by looking to a set of already observed examples. The generated procedure will be able to correctly classify these samples of examples; but especially to have a good predictive capacity to correctly classify new cases.

Methods used by learning systems are very numerous and result from a variety of scientific disciplines. The statistical classification methods assume that descriptions of objects in a given class are divided according to a specific structure of the class. These methods makes priori assumptions on the distributions of features in the context of classes and the classification procedure will be built using probabilistic

assumptions like in Bayesian classifiers. The variety of methods comes from diversity of possible assumptions. These methods are called sub-parametric. Non-parametric methods (without priori assumption on distributions) were also proposed in statistics. Methods resulting from artificial intelligence are non-parametric methods. We distinguish the symbolic methods (the produced classification procedure can be written in the form of a set of rules), the non-symbolic methods or adaptive methods (the produced classification procedure is like a "black box"). Among the symbolic methods, the most used are based on the decision trees and rule-based learning. For the adaptive methods, we distinguish two big classes: neuron networks and genetic algorithms.

In this thesis, we first investigate a primary reflection on symbolic classification methods especially rule-based learning before going to a *deep* study on statistical classification methods which constitute our principal contribution in this work. The interest accorded to rule-based learning is motivated by more than one reason: in fact, most of existing expert systems are rule-based since they use symbolic rules of the form if-then rules in their knowledge representation language. This is due to the very important benefits that production rules offer to knowledge representation and reasoning in expert systems, such as naturalness, modularity and ease of explanation.

Although their advantages, in terms of expressivity power, serious problems could be identified when learning rules. First, it is known that the search process for a set of rules that optimize the cost function is the main drawback of rule induction methods because this searching is considered as a difficult optimization problem. Second even after learning these rules, different problems can occur: especially the multiple covering case where a test example could be covered or classified by more than one rule having different labels. The non-covering case is also a conflicting situation in which no rule can cover the test example.

In this work, we first focus to review the most known rule based classifiers in the literature and then test the ability of using fuzzy rules instead of crisp ones to overcome some previously cited problems. In this context, we propose a Possibilistic Rule based Classifier (PRC) which is mainly based on the learning process of the well known PART algorithm proposed by Frank and Witten (1998) [85] and aims to extend and modify this algorithm in order to improve its efficiency. The PRC differs from the PART algorithm mainly in three ways. In particular, this classifier investigates fuzzy rules instead of crisp rules to give more flexibility to rule decision boundaries. Moreover, it considers rules in unordered manner instead of decision lists at deduction time. The PRC is mainly based on a Possibilistic Rule-based Reasoning which enables computing the relevance possibility of each rule to a given test example and thus considering rules in a symmetric way. Finally in the PRC, we propose to use a rule fuzzy extension with large supports to cover non-covreing examples instead of a prefixed default class.

The objective of this part is thus to test the ability of the proposed PRC to ameliorate the classification ability of a classical rule-based classifier. In fact, we aim to measure the effect of rule fuzzification on the performance of a classical classifier.

When we deal with classification, we are faced to another serious problem which is related to the *quality* of data to classify. Most of existing classification techniques assume that the available set of examples, from which a classifier will be trained, is perfect which is not always realistic. In many real world problems, data could be pervaded with imperfection (imprecision or uncertainty). So it is important for any information system to be able to deal with such type of data if it attempts to provide a complete and accurate model of reality. However, this task is hardly achieved by classical classification techniques for more than one reason: i) first it is difficult to understand the various aspects of imprecision and uncertainty and ii) in practice there aren't standard imperfect benchmarks which could be used in classification as in the perfect case.

In real classification problems, imperfect information is ubiquitous, almost all of the information we have about the real world is imperfect. *Incompleteness*, *imprecision* and *uncertainty* are the very common situations of imperfect data. A collected data set may include examples with some missing attributes. Incompleteness in attributes may be caused by failures of sensors or bad maintenance of the data set. On the other hand, data may contain training examples with imprecise and/or uncertain attributes or class labels. In such situation, attributes or classes might be labeled in a vague way by a subset of values or by assigning a degree of belief for each possible label.

For a long time, almost all aspects of imperfect data were modeled by probability theory but in the last 40 years, many new models, such as fuzzy set theory [173], evidence theory [49] [156] and possibility theory [63] have been developed to represent imperfection in data. The large number of models reflects the recent appreciation that there exist many aspects of imperfection. Probability theory, as good as it is, is not the unique normative model that can deal with all kinds of imperfect data. As discussed in Dubois et al. [56], this theory is limited in the sense that it cannot distinguish between *total ignorance* and *ambiguity*.

The subject of this thesis enters within the framework of uncertainty treatment in the classification of numerical data. Possibilistic classifiers are recent approaches, based on possibility theory, to study the classification problem. There are only few works that treat the problem of possibilistic classification in the literature, although possibility theory is a suitable tool for the treatment of imperfect knowledge. It should be noted that most of existing *Naïve Possibilistic Classifiers* deal only with categorical attributes and require a supplementary discretization phase for numerical data. Possibilistic classifiers, which are mainly based on naive possibilistic networks, have similar architecture than that of Bayesians networks known by their simplicities

and effectiveness for classification.

The aim of this work is to investigate possibilistic classifiers and to discuss their ability to deal with perfect and imperfect *numerical* data. These classifiers can be viewed as a natural counterpart of Naive Bayesian Classifiers (NBCs) mainly based on probabilistic distributions. The study of possibilistic classifiers is motivated by the good performance of NBCs and by the ability of possibility theory to handle poor and imperfect data. For this reason in the second part of this thesis we are interested to develop and test a variety of possibilistic classifiers which are appropriate to the perfect or imperfect data.

In this thesis, we propose and study two families of possibilistic classifiers for numerical attributes which constitute our principal contribution. The first family, adopting the normality assumption of data distributions, is based on a method of transformation from probability to possibility allowing transforming traditional Naive Bayesian Classifier (NBC) into a Naive Possibilistic Classifier (NPC). The advantage of this transformation is to add more tolerance in the description of the classes. We also analyze the feasibility of Flexible Naive Possibilistic Classifier (FNPC) which constitutes a possibilistic counterpart of the Flexible Naive Bayesian Classifier (FNBC). In the same family, we also study other variants of Gaussian based Possibilistic classifiers denoted NPC-2 and FNPC-2 which are based on computing a possibility distribution as a family of Gaussian distributions for which the parameters are in a chosen confidence interval.

The second family of possibilistic classifiers gives up the normality assumption and reflects a direct representation of data. We propose two classifiers denoted Fuzzy Histogram Classifier (FuHC) and Nearest Neighbor-based Possibilistic Classifier (NNPC). The first one exploits an idea of proximity between attribute values in an additive manner, while the second one is based only on analyzing proximities between attributes without counting them. We also develop an approach improving the performance of possibilistic classifiers in the case of undistinguishable classes where class plausibilities are too close. We propose to exploit a nearest neighbor approach to separate undistinguishable classes.

Our last contribution in this thesis consists to extend possibilistic classifiers, proposed for numerical data, to support uncertainty in data representation. Indeed, we intend to deal with two types of uncertainty: uncertainty related to the class attribute in the training set modeled through a possibility distribution over class labels and uncertainty related to attribute values in the testing set represented through intervals for continuous data. We first adjust possibilistic classification model, previously proposed for the certain case, to support uncertainty in class labels. Then, we propose an extension principle based algorithm to deal with uncertain attribute values.

This thesis is organized in the following six chapters:

- *Chapter 1: Possibility theory for uncertainty treatment.* This chapter offers an overview of essential concepts of possibility theory and a brief presentation of other related uncertainty theories such as probability theory, belief function theory and fuzzy set theory. A discussion and links between possibility theory and probability theory is given at the end of this chapter.

- *Chapter 2: Classification Methods.* In this chapter we have briefly presented the most used classification techniques and we focused more on Bayesian classifiers. Advantages and limits of each method are also studied. In this dissertation, we are mainly interested to statistical methods based on Bayesian networks.

- *Chapter 3: Rule-based learning.* This chapter gives an overview of the most known rule-based learning. Two families of methods are pointed out: the method based on sequential covering algorithm and methods based on decision tree for rule generation. Other hybrid approaches exploiting a combined process from the two previously methods are also presented mainly the PART algorithm. Finally experiments for the proposed approaches has been reported.

- *Chapter 4: A Possibilistic Rule based Classifier.* In this chapter we propose a family of Possibilistic Rule-based Classifiers (PRCs) which uses a Possibilistic Rule-based Reasoning to deal with the multiple classification and non-covering problems. The PRC is an extension and a modification of the PART algorithm which keeps the same rule learning step as the PART and differs mainly in the type of rules it represent, the manner to consider rules for deduction and the choice of the default class.

- *Chapter 5: Possibilistic Classifiers for perfect/imperfect numerical data:* This chapter investigates a possibilistic classification paradigm that may be viewed as a counterpart of Bayesian classification and that applies to continuous attribute domains. For this purpose, we have proposed two families of possibilistic classifiers: the first family called, Gaussian-based Possibilistic Classifiers. The second family of possibilistic classifiers abandons the normality assumption and has a direct representation of data. As an attempt to improve the performance of possibilistic classifiers, we have proposed a hybrid classification method that is based on a Nearest Neighbor Heuristic used for separating classes having close plausibility estimates. In this chapter we also extend the possibilistic classifiers to handle uncertainty and imprecision in input data sets.

- *Chapter 6: Experimenting Possibilistic Classifiers for perfect/imperfect data.* This chapter presents all necessary elements which are needed to experiment all the proposed possibilistic classifiers presented in Chapter 5. Data sets have been presented and the way these data sets have been contaminated by uncertain class labels and attributes is detailed. Then, different performance criteria have been

proposed for the evaluation of possibilistic classifiers. Results of the application of the proposed approaches on different perfect/imperfect data sets have been reported and analyzed.

Finally, a general conclusion summarizes the major achievements of this thesis and presents possible future developments.

An appendix shows some graphical interfaces of the toolbox that we have developed to implement the PRC approaches as well as all Possibilistic Classifiers i.e. the NBC, the FNBC, the NPC, the NPC-2, the FNPC, the FNPC-2, the FuHC, the NNPC, and the NNH.

Chapter 1

Possibility theory for uncertainty treatment

1.1 Introduction

Most of existing methods for data analysis have been developed to treat perfect numeric data which are usually supposed to be valid, complete and specially prepared for a special purpose. However in practice, statistician are more and more faced to data whose nature and format do not correspond to this traditional scheme.

There are many reasons which cause the appearance of imperfection in data representation. In particular, we note the development of exponential means of data recording and storage. Those are not systematically validated and formatted for a traditional analysis. Besides, it seems illusory to consider an absolute precision of the data. Any measurement system has its limits in terms of precision estimated by its error risk. Thus instead of masking imprecision in real problems dealing with numerical data (by considering only the median value for example), it can be more interesting in contrary to integrate the imprecise knowledge in the analysis.

Linguistic descriptions provided by experts, which are usually vague, uncertain and/or imprecise are another common forms of data not easily analyzable in a traditional manner. In fact, imperfection can be caused by the difficulty for a expert to express his knowledge about a situation to which he likes to make decision. Indeed the precise qualification of a situation by an expert may be difficult in some domains (e.g., in medical diagnosis, or in law application data) and may be also costly.

Since imperfect knowledge-bases will always exist in practical systems, it is usual for a system has to be able to model and deal with data base imperfection and uncertainty. In practice, it is expected that ignoring a partial knowledge about a situation and considering it as unknown is not faithful and may conducts to erroneous

decision making. Thus, the solution is to find suitable tools for modeling and dealing with such imperfect knowledge.

For a long time, probability theory was the classical most used tool. In the last years, several non-classical theories of uncertainty have been proposed in order to deal with uncertain and imprecise data. The most known are evidence theory [156], fuzzy set theory [173], possibility theory [174][63], imprecise probabilities [165].

Even if we briefly introduce a panoply of theories for data imperfection in this chapter, we are mainly interested to discuss basic foundation of possibility theory. This last is known by its simplicity and its ability to be applied in different areas such as information fusion, machine learning, default reasoning and diagnostics...

This chapter is organized as follows: in the next section we distinguish between different types of imperfect data mainly imprecision and uncertainty. Section 3 provides basic theories dealing with data imperfection. In Section 4, we are totally interested to give basic notions in possibility theory. Section 5 gives a comparative study between possibility and probability theory for handling uncertainty in data.

1.2 Imperfection in data

In many domains, numerical data is assumed to give a true representation of reality such as in pattern recognition, data sensor based systems and data analysis. However in general, such data may be pervaded with imperfection: imprecision, uncertainty, ambiguity and incompleteness...

Several authors are interested to precisely analyze different forms of imperfection in data. We cite for example Bonissone and Tong [15], Bosc et al.[19], Smets [157], Dubois and Prade [73]. In this section, we give a brief presentation of the topology of imperfect data. One can decompose imperfection in data mainly into three categories (nonexclusive): uncertainty, inconsistency and imprecision, each one can be divided into several sub-categories [128] (Figure 1.1).

To illustrate these concepts, let us consider the example of *election for the Tunisian Constituent Assembly* hold on 23 October 2011. For a particular agent, an observer or a journalist for example, we consider that there are a lot of unknown information that should be defined such as the *name of each politic Party (P_1, P_2,...)*, *the elected Parties*, and *the number of members in the Tunisian Constituent Assembly for each Party*.

We note that *incompleteness* can be considered as a particular case of *imprecision* and refers to a totally absence of information. For example an observer do not actually has the election result. In contrary, if the observer only know that elected

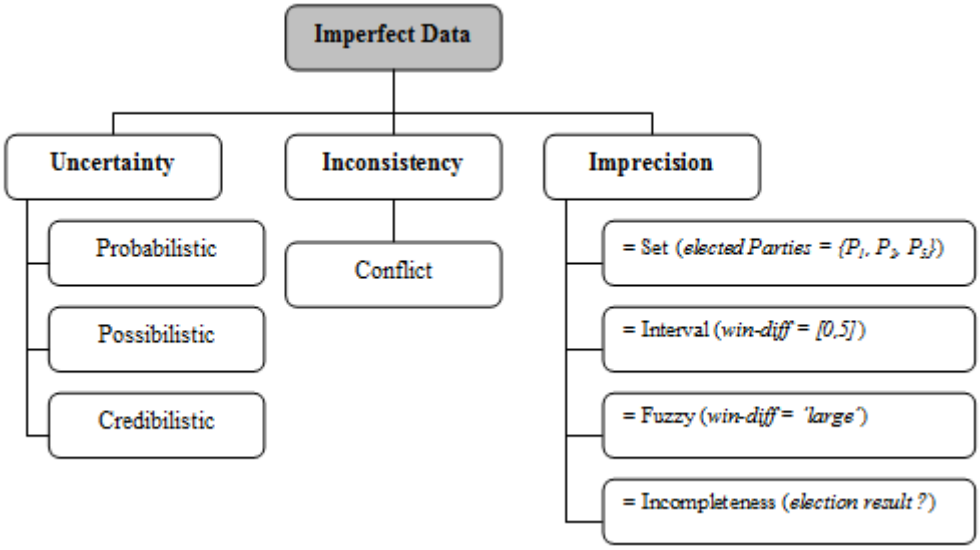

Figure 1.1: Different forms of imperfect data

Parties will be P_1, P_2 or P_5 or that P_1 wins P_2 on at least five members, this available information will be considered as *imprecise*. For the categorical feature, only a set of different issues is given to describe this imprecise information, i.e: *Elected Parties* $\in \{P_1, P_2, P_5\}$. For the numerical feature, we know that its value belongs to a certain interval, i.e: $Win - difference \leq 5$.

Other forms of imprecision could also be considered in the same context. Data could be represented in a *vague* and/or a $fuzzy$ manner if this data is announced in a linguistic form. For example, one can say that Party P_1 gains the "major" members in the Constituent Assembly or that P_2 wins P_5 with a "large" difference of members. These information are considered as vague since no unique significance can be given for "major" or "large" terms and thus they refers to a set of values whose boundaries are not precisely defined.

Uncertainty refers to the veracity of the information which describes the state of knowledge of an observer. For example if information misses to a journalist, it can think to acquire a politician to give him the expected number of elected members for the Party P_1. This journalist should takes into account the credibility of this politician since even if the given information is precise and complete it could be erroneous.

Some authors choose to distinguish three types of uncertainty:

- *Objective* uncertainty that one can randomly assimilate (one can use sensor networks which may have some error risks such as a temperature or pelvimetry sensors).
- *Subjective* uncertainty primarily related to *credibility* that one assigns to an

information source which provides the information (the journalist should distinguish reliable politician.

- *Epistemic* uncertainty which is related to the lack or the incompleteness of the information [75].

In numerical case, the precision concept could be represented by a set of points in the space of possible states. The more this set tends to the singleton, the more the information is precise. Imprecision occurs when the desired information is only a simple element, whereas the available information is sub-set containing the element we search. The certainty is a more delicate concept to represent. It constitutes an appreciation on the veracity of the obtained information: it is a concept different from imprecision. Generally the doubt is evaluated by a level on a scale from 0 to 1, level 0,5 corresponds to the maximum uncertainty.

These types of uncertainty can be modelled in more than one way:

- Probabilistic uncertainty: what is the success chance of *Party* P_1 in the next election,

- Possibilistic uncertainty: the possibility that *Party* P_1 wins more than 3 members,

- Credibilistic uncertainty: my own belief that *Party* P_1 will gain.

To these two forms of imperfection, Bosc et al.[19] adds the *inconsistency* which occurs in presence of redundancy when many information are in conflict. For example, if we consider two journalists asked to give the total number of electoral lists for a given Party. The first one say that there is five lists, the second say that it exceeds seven lists. This inconsistent information can occur when the two journalists consult two different information sources and one of them is outdated for example.

It is seen that data imperfection can take several nonexclusive forms. For a long time, it is considered that the probabilistic framework is the only suitable framework for representing and handling imperfect data. In the thirty last years, others theories for imprecision and uncertainty management have been considered as alternative to the probabilistic framework. This is due to the fact that, in many real world problems, imperfection could not always be explained by a random phenomenon. So the probabilistic vision seems to be unsatisfactory.

The following sections report basic notions of these theories and also a comparative study between them.

1.3 Theories dealing with imperfect data

In this section we briefly recall basic notions of the main used theories dealing with uncertainty in the literature. For this purpose, we will present probability, evidence, fuzzy set and possibility theories. In the following we also give some notations useful for this presentation:

- Let denote Ω the universe of discourse for the given problem with $\Omega = w_1, w_2, ..., w_n$.
- 2^Ω the set of all possible subsets of Ω.
- A is an element of 2^Ω or a subset of Ω .

1.3.1 Probability theory

Probability theory is the most traditional model to deal with uncertain information. It dates from the 17th century. It is mainly a quantitative tool for uncertainty treatment which often touches several real world domains (management, economy, sciences, industry, etc...). Whereas for several people the probability means a quantitative (based on frequency) process of uncertainty treatment, for others it is rather a Game theory or a randomness theory which is an indispensable tool to model any real world problem [62].

A probability measure p on a finite space Ω is a non-negative mapping $p : \Omega \rightarrow [0, 1]$ such that $\sum_{x\in\Omega} p(x) = 1$. A subset $A \subseteq \Omega$, is called a random event, and for a given p, the probability measure P of the event A is $P(A) = \sum_{x\in A} p(x)$. This measure evaluates to what extend this event could happen in term of likelihood which is simply the total mass that was assigned collectively to elements of A. Note that the probability $P(A)$ of the event A corresponds to the expectation of the function of A which is such that it takes values 1 on elements $x \in A$, and zero on elements $x \in A^c$, with A^c is the complement of the event A.

The probability of the event A could have a purely classical $frequency$ based interpretation usually called *objective probability* which is described by the quotient of the number of favorable issues (realizing A), N_A, on the number of possible issues N, this quotient requires that the number of possible issues is finite.

In the other hand, probability could be interpreted as the degree of confidence that someone assigns to the occurrence of an event A. This is the case of *subjective probablity* which basis concepts have been introduced by Finetti [81], Savage [154] and Berenji [11]. The probability interpretation proposed by L.J. Savage is classically called "subjective"; it considers that a probability value on an alternative set is given

as a feeling representation of uncertainty in respect to an eventual choice between several eventualities [154][78].

Formal probability theory is essentially based on the three Kolmogorov axioms and deduced properties:

Axiom 1.1. $\forall A \subset \Omega, P(A) \geq 0, P(A) \leq P(\Omega)$

Axiom 1.2. $P(\emptyset) = 0,$ *and* $P(\Omega) = 1$

Axiom 1.3. $\forall A, B \subset \Omega, P(A \cup B) = P(A) + P(B) - P(A \cap B)$

A and B are considered as *independent* events if $P(A \cap B) = P(A) \times P(B)$. From the three axioms we can conduct the following properties [153].

Property 1.1. $\forall A \subset \Omega, 0 \leq P(A) \leq 1$

Property 1.2. $\forall A \subset \Omega, \sum_i P(A_i) = 1$

Property 1.3. $\forall A \subset \Omega, P(A) + P(\bar{A}) = 1$

Given a probability distribution $p : \Omega \rightarrow [0, 1]$ on element w of event A, a probability measure can be deduced from the additive property (1.2) in the discrete domain:

$$\forall A \subset \Omega, P(A) = \sum_{w \in A} p(w), \tag{1.1}$$

In the continuous domain, the function p is a probability density which is totally defined by its cumulative distribution $P : R \rightarrow [0, 1]$:

$$\forall w \in R, \int_{\Omega} p(w)dw = 1, \text{ and } P(A) = \int_{A} p(w)dw \tag{1.2}$$

Conditional probability

Assigning a probability measure to an event A, mainly in the subjective case, is not carried out definitively. This assignment can be justified in a certain context which is limited by the available knowledge base of the expert. If then later this expert obtains new pieces of information, this leads to change the knowledge base and thus update these probabilities.

The Bayes reasoning is a continuation of classical probabilistic reasoning which enables modeling uncertain situation. Bayes rule can be defined as a model by which the expert can update his belief about an event A using conditional probabilities.

Conditional probability P(A/B) estimates the probability that an event A occurs if we know that event B has happen. This probability is formulated by Bayes in the following:

$$\forall A, B \subset \Omega, P(A/B) = \frac{P(A \cap B)}{P(B)}, P(B) \neq 0 \tag{1.3}$$

In this context, the knowledge base can simply be defined by all probabilities P(A/B) for all events A and B. Then for a current situation, an expert only selects the appropriate conditional probability based on the available knowledge. Conditional probability can be viewed as a revising probability measure based on the arrival of new knowledge. The quantity P(A/B) is then considered as the new probability of A when the expert hears that event B occurred [73]. The principle of minimal change is the basic for belief revision: the expert minimally revises his/her beliefs so as to absorb the new information item without violating property 1.2. [73].

Bayes Theorem

Bayes' Theorem is a theorem of probability theory originally stated by the Reverend Thomas Bayes and it is used for determining conditional probability. The theorem provides a way for understanding how the probability that a theory is true is affected by a new piece of evidence. It has been used in a wide variety of contexts, for example in finance, Bayes' Theorem can be used to rate the risk of lending money to potential borrowers.

Let's consider the definition of conditional probability: $P(A \cap B) = P(A/B) * P(B)$. But we can switch the roles of A and B: $P(A \cap B) = P(B/A) * P(A)$. Equating the right hand sides, we obtain: P(B/A)* P(A) = P(A/B)*P(B) which leads to the following Bayes'Theorem:

$$P(A/B) = \frac{P(B/A) * P(A)}{P(B)} \tag{1.4}$$

In problems related to probabilistic inference, we are often trying to estimate the most probable underlying model for a random process, based on some observed evidence. If A represents a given set of model parameters, and B represents the set of observed data values, then the terms in Equation 1.4 are given the following terminology:

- P(A) is the prior probability of the model A (in the absence of any evidence);
- P(B) is the probability of the evidence B;

- P(B/A) is the likelihood that the evidence B was produced, given that the model was A;
- P(A/B) is the posterior probability of the model being A, given that the evidence is B.

Imprecise probability

Assigning precise probabilities for events can be seen as limiting in some contexts. In fact, the probability theory cannot adequately model any type of uncertainty because probability measures are too precise such that they do not reflect domain reality. Precise probabilities are limiting in more than one sight:

- When dealing with objective probabilities, it can be happen that frequency cannot be precisely defined. At the limit, we can only know that this event frequency belongs to an interval [166].

- In general, there is rarely sufficient information about the data to help choosing the convenient probability distribution which best estimates the true distribution of data.

- In a probabilistic model the estimated parameters, such as the mean or the standard deviation, could be irrelevant when few samples are available.

- The indifference principle, used in probability theory, seems to be the most point which should be discussed. This principle assumes that in the case of total ignorance, it is recommended to say that all possible events are equi-probable. This subjective probability theory seems to be not very adapted to situations where knowledge is poor [62].

- Walley [165] proposes the idea of exchangeable bets and thinks that it is more natural, for an expert, to give prices for example as an interval by providing maximal baying and minimal selling prices for gambles instead of giving precise values. These probability intervals allow some imprecision in the expectation provided by experts. Other justifications can be found in Walley [165]. Upper and lower expectations defines a closed convex sets of probabilities also called *credal sets* [73].

Imprecise probability is modeled by a family $\wp$ of probability distributions used to generate uncertainty models instead of using only one particular probability distribution. Lower and upper probability bounds are defined as follows:

$$P_*(A) = \inf_{P \in \wp} P(A) \; and \; P^*(A) = \sup_{P \in \wp} P(A) \tag{1.5}$$

P_* and P^* are usually called *lower* and *upper envelopes* [165]. These two measures are dual to each other i.e. $P_*(A) = 1 - P^*(\neg A)$ and the probability family is completely defined if one of the two measures is given. Moreover, the interval $[P_*(A), P^*(A)]$ represents the expert ignorance about the event A. When this interval become too width, we can say that it represents total ignorance and when it is reduced to only one point it represent a precise probability.

1.3.2 Belief function theory

Belief function theory is initially named *evidential theory* and also *Dempster-Shafer* theory since it is developed by Dempster (1967)[49] and Shafer (1976)[156]. This theory is a robust tool for representing uncertain data. We give here some basic notions of this theory.

Given Ω the universe of discourse, a mass function is defined on the set of all possible sub-sets 2^Ω and affects to each sub-set a value in [0, 1] representing its elementary belief mass. This later enables modeling uncertain and imprecise knowledge given by an expert (a source, a classifier,...). More formally, a *basic belief assignment* (bba) is defined as follows:

$$m : 2^\Omega \rightarrow [0, 1]$$

$$m(\emptyset) = 0 \text{ and } \sum_{A \subseteq \Omega} m^\Omega(A) = 1$$

The mass $m(A)$, assigned to the element A and usually called *basic belief mass* (bbm), represents the elementary belief degree of the expert that the real value of a variable is included or equal to A. A sub-set A having a not null belief mass is a said to be a $focal\ element$.

Although condition $m(\emptyset) = 0$ is not always necessarily, it helps to get bba normalization. This condition supposes that we are in a closed domain Ω. If $m(\emptyset) \neq 0$, this mass can be interpreted as the degree of belief that the desired value is a non enumerated hypothesis in Ω. In a closed domain, Ω is assumed to be exhaustive which means that all possible hypothesis are enumerated in Ω and thus requires a null mass to be assigned to the empty set.

Smets and Kennes (1994)[159] claims that an open domain Ω is assumed to be non exhaustive so a non null mass could be assigned to the empty set.

The belief function (or credibly function) *bel* represents the minimal belief degree affected to a sub-set of 2^Ω which is justified by available information. $bel^\Omega(A)$ measures the degree for which the information $B \subseteq A$, given by a source, supports A. The belief function *bel* is defined as follows:

$$bel^{\Omega} : 2^{\Omega} \rightarrow [0,1]$$

$$A \mapsto \sum_{B \subseteq A, B \neq \emptyset} m(B)$$

The plausibility function *pl* quantifies the belief degree assigned to propositions not contradictory with sub-set A. It is the maximum belief degree that could be given to A:

$$pl^{\Omega} : 2^{\Omega} \rightarrow [0,1]$$

$$A \mapsto \sum_{A \cap B \neq 0} m(B))$$

The plausibility function is a dual value to the belief function:

$$pl(A) = 1 - bel(\overline{A})$$

Combination rules

Belief function provides a robust tool for combining different knowledge, represented by mass functions defined on the same domain and given by different sources. Combination enables confronting these different knowledge in the aim to obtain a unified one used for decision making.

There are many different combination rules, we present here only the most used rules. A more complete survey on these rules can be found in Sun and Farooq (2004)[163]. The *conjunctive combination rule* (CCR) is used when information sources are fully reliable. The *conjunctive* rule enables to combine two distinct mass functions m_1^{Ω} and m_2^{Ω} as follows:

$$(m_1^{\Omega} \cap m_2^{\Omega})(A) = \sum_{B \cap C = A} m_1(B) \times m_2(C)$$

When one or more sources are non reliable, it is recommended to use the *disjunctive combination rule* defined by [158]:

$$(m_1^{\Omega} \cup m_2^{\Omega})(A) = \sum_{B \cup C = A} m_1(B) \times m_2(C)$$

1.3.3 Fuzzy set theory

As introduced in the beginning of this chapter, developed systems should be able to deal with data imprecision and uncertainty often involved in real world problems. Classical mathematical models usually make restrictive hypotheses that require data to be strongly precise.

Let us take the example of air-conditioning: if we want to obtain a "fresh" temperature, we want to know which range of temperature will be appropriate (the request is obviously vague); moreover the reliability of sensors should be taken into account (the measurement of the room temperature is uncertain). Linguistic variables like "fresh", "hot", or "cold" have different interpretations which differ from one individual to another. In addition, the treatment of such data is attached to uncertainty.

In order to be able to represent such types of information, Zadeh proposed to model the mechanism of human thought using an approximate reasoning based on linguistic variables. He introduced the *fuzzy set* theory in 1965 [173], which generalizes classical set theory. More generally, the term of *fuzzy logic* corresponds to all the developments resulting from the fuzzy set theory.

Many applications have been developed in several domains where any deterministic model could not be applied like situations for which data imprecision returns impossible the control using traditional methods.

Contrary to the boolean logic, fuzzy logic allows to a variable to be in another state than "true" (1) or "false" (0). There are, rather, degrees of checking for each variable.

Example 1.1.

To illustrate the contribution of fuzzy sets if compared to boolean logic, let us consider the example of vehicle speed on a trunk road. Normal speed is of 90 km/h. A speed can be seen as high above 100 km/h, and as not at all high under 80 km/h. Boolean logic would consider that the speed will be at 100% high starting from 100 km/h and low otherwise. In the opposite, fuzzy logic allows associating a function degree $\in [0, 1]$ for each element w in the fuzzy set A. In this fuzzy extension, the speed is seen as not at all high only for the lower part of 80km/h. One can say that, in the lower part of 80 km/h, speed is high to the degree 0. Speed is seen as surely high (with the degree 1) above 100 km/h. The speed is thus high at the degree 0.5 for 90 km/h, and at the degree 0.25 for 85 km/h (Figure 1.2).

After this brief introduction, let us give now some basic notions in fuzzy set theory.

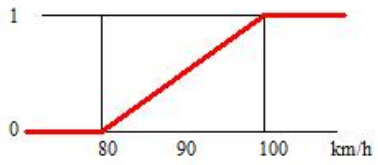

Figure 1.2: Vehicle speed example

A *fuzzy set* A of Ω is totally defined by a membership function which corresponds to each element $w \in \Omega$ the degree $\mu_A(w) \in [0, 1]$.

Definitions

The following basic notions characterizes the fuzzy set A (Figure 1.3):

- *Support(A)*: $\{w \in \Omega : \mu_A(w) \neq 0\}$,
- *Core (A)*: $\{w \in \Omega : \mu_A(w) = 1\}$,
- *Height*: $h(A) = Sup_{w \in \Omega} \mu_A(w)$,
- *Normalized fuzzy set*: A is normalized if $h(A) = 1$,
- *Cardinality*: $\mid A \mid = \sum_{w \in \Omega} \mu_A(w)$,
- *The* $\alpha - cut$ *of* A denoted A_α is the set of elements in A whose membership function $\mu_A(w) \geq \alpha$.

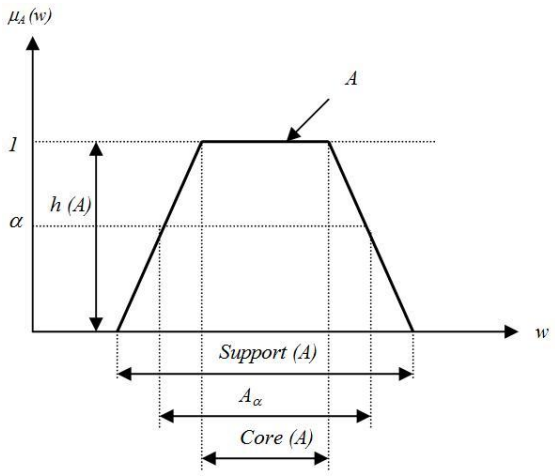

Figure 1.3: A membership function

Fuzzy operators

Fuzzy subsets are used to describe vague/ imprecise concepts, gradual properties or uncertain events. Various properties, initially appropriate for classical set theory, are extended in the context of fuzzy sets. In particular, inclusion, union, intersection, complement, relation and convexity are established to such subsets. Let A, B, C and D be fuzzy sets:

1. *Equality*: $A = B$ if $\forall w \in \Omega : \mu_A(w) = \mu_B(w)$
2. *Inclusion*: $A \subset B$ if $\forall w \in \Omega : \mu_A(w) \leq \mu_B(w)$
3. *Intersection*: $A \cap B = C$ such as $\forall w \in \Omega : \mu_C(w) = min(\mu_A(w), \mu_B(w))$
4. *Union*: $A \cup B = D$ such as $\forall w \in \Omega : \mu_D(w) = max(\mu_A(w), \mu_B(w))$
5. *Complement*: $\forall A \in \Omega : \mu_{A^c}(w) = 1 - \mu_A(w)$

Fuzzy concepts

- A fuzzy sub-set is said to be *convex* if its membership function is convex.
- The *cartesian product* of two fuzzy sets A_1 and A_2 is defined by:

$$A = A_1 \times A_2, \forall w = (w_1, ..., w_n) \in \Omega; \mu_A(w) = min(\mu_{A_1}(w_1), ..., \mu_{A_n}(w_n))$$

- The *projection* on Ω_1 of a fuzzy set $A \in \Omega_1 \times \Omega_2$ is the fuzzy set B with membership function defined by:

$$\forall w_1 \in \Omega_1, \mu_B(w_1) = Sup_{w_2 \in \Omega_2} \mu_A(w_1, w_2)$$

Extension principle

Zadeh in [173] introduced the *extension principle*, one of the most important issues in fuzzy set theory, which allows to exploit our classical knowledge in the case of fuzzy data (fuzzy arithmetic, fuzzy relations...).

Let F be a real function such that $F : \Omega \to R, R$ being the set of real numbers. Let $F(w) = u$ and let $\mu_A(w)$ be membership for w.

Assume that A is a fuzzy set defined in Ω, using the extension principle, the membership for u is:

$$\mu_B(u) = sup\{\mu_A(w) | F(w) = u\}. \tag{1.6}$$

The extension principle defines a fuzzy set B in R which is the direct image of A in the intermediate of F.

1.4 Basic notions in possibility theory

Possibility theory [63] has been introduced by Zadeh [174]. It handles epistemic uncertainty in a qualitative or quantitative way. In particular, possibility theory is suitable for the representation of imprecise information. For a more complete introduction to possibility theory, see [70]. Possibility theory deals with possibility degrees which is rather employed as a graded notion much in the same way as probabilities. Before going to more technical aspects, we first give the basic underlying the notion of *possibility*.

As with probability theory, which can produce different interpretations (frequentist or subjective sense), in possibility theory, possibilistic degrees may support several interpretations [69]. The most used senses of "possibility" are the $feasibility$ or the *degree of ease* to achieve an action. A logical interpretation of "possibility" is that of *consistency* of a given event with the available knowledge which evaluates to what degree this event is not contradicting with what we know.

In this quantitative setting, possibility distributions have two types of interpretations. The first one, that is related to fuzzy set theory, is the description of gradual properties. For instance, the definition of linguistic expressions such that "long", "old" or "expensive" does not refer to a specific value, but to a set of possible values in a specific context. For instance, a possibility distribution may describe the concept "expensive" for a house in a particular area. In such a case, each price will be associated with a possibility degree which quantifies how much this price is typical with respect to the concept "expensive". Assigned to events, possibility degrees can also represent *plausibility* which reflects the belief degree of the expert that a certain event will occur.

1.4.1 Possibility distributions

Possibility theory is based on *possibility distributions*. Given a universe of discourse $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$, a possibility distribution π is a function that associates to each element ω_i from the universe of discourse Ω a value in a bounded and linearly ordered valuation set $(L, <)$. This value is called a *possibility degree*. This scale may be quantitative, or qualitative when only the ordering between the degrees makes sense. In this thesis, we are only interested to the *quantitative* reading of possibility degrees and L is taken as the unit interval [0, 1]. A possibility distribution is used as an

elastic constraint that restricts the more or less possible values of a single-valued variable.

In this scope, a possibility distribution is viewed as a family of probability distributions (see [55] for an overview). Thus, a possibility distribution π represents the family of the probability distributions for which the measure of each subset of Ω is bounded by its necessity and its possibility measures.

By convention, $\pi(\omega_i) = 1$ means that it is fully possible that ω_i is the value of the variable. Note that distinct value ω_i, ω_j may be such that $\pi(\omega_i) = 1 = \pi(\omega_j)$. $\pi(\omega_i) = 0$ means that ω_i is impossible as the value of the variable. Thanks to the use of the interval [0,1], intermediary degrees of possibility can be assessed, which enable us to acknowledge that some values are more possible than others. In possibility theory, different important particular cases of knowledge situation can be represented:

- Complete knowledge: $\forall \omega_i, \pi(\omega_i) = 1$ *and* $\forall \omega_i \neq \omega_j, \pi(\omega_j) = 0$.
- Partial ignorance: $\forall \omega_i \in A \subseteq \Omega, \pi(\omega_i) = 1$, $\forall \omega_i \notin A, \pi(\omega_i) = 0$ when A is not a singleton.
- Total ignorance: $\forall \omega_i \in \Omega, \pi(\omega_i) = 1$ (*all values in* Ω *are possible*).

1.4.2 Possibility and Necessity measures

A possibility distribution π on Ω enables events to be qualified in terms of their plausibility and their certainty, by means of two dual possibility and necessity measures that are respectively defined for an event $A \subseteq 2^{\Omega}$ by the formulas:

$$\Pi(A) = max_{\omega \in A} \pi(\omega) \tag{1.7}$$

$$N(A) = min_{\omega \notin A}(1 - \pi(\omega)) = 1 - \Pi(\overline{A}) \tag{1.8}$$

$\Pi(A)$ evaluates to what extent A is consistent with our knowledge represented by π. Indeed, the evaluation provided by $\Pi(A)$ corresponds to a degree of *non-emptiness of the intersection* of the classical subset A with the fuzzy set having π as membership function. Moreover, $N(A)$ evaluates to what extent A is certainly implied by our knowledge, since it is a degree of *inclusion* of the fuzzy set corresponding to π into the subset A.

Quantitative possibility distributions can represent, or more generally approximate, a family of probability measures [66]. Indeed, a possibility measure Π can be viewed as an upper bound of a probability measure, and associated with the family of probability measures defined by:

$$\mathcal{P}(\Pi) = \{P \text{ s. t. } \forall A, \Pi(A) \geq P(A)\}$$

Thanks to the duality between Π and N and the auto-duality of P ($P(A) = 1 - P(\overline{A})$), it is clear that:

$$\forall P \in \mathcal{P}(\Pi), \forall A, \Pi(A) \geq P(A) \geq N(A).$$

This is the starting point for defining a probability-possibility transform. The width of the gap between $N(A)$ and $\Pi(A)$ evaluates the amount of ignorance about A, since it corresponds to the interval containing the imprecisely known probability. Thus, possibility distributions can in particular represent precise or imprecise information (representable by classical subsets) as well as complete ignorance. The possibilistic representation of complete ignorance should not be confused with a uniform probability distribution. Indeed, with the above representation, we have $\Pi(A) = 1$ for any non empty event A, and $N(A) = 0$ for any event A different from Ω, while a uniform probability distribution on a universe with more than two elements associates non trivial events with a probability degree strictly between 0 and 1, which sounds paradoxical for a situation of complete ignorance. Possibility theory is particularly suited for representing situations of partial or complete ignorance (see [55], [73] for detailed comparative discussions between probability and possibility).

1.4.3 Conditional Possibility and Possibilistic Bayesian Rule

Conditioning in possibility theory is defined through a counterpart of Bayes rule, namely

$$\Pi(A \cap B) = \Pi(A|B) * \Pi(B)$$

It is has been shown that there are only two basic choices for $*$, either minimum or the product [65]. The min operator is suitable in the qualitative possibility theory setting, while the product should be used in quantitative possibility theory [40].

Thus, possibilistic conditioning corresponds to revising an initial possibility distribution π, when a new information $B \subseteq \Omega$ is now available.

- Qualitative setting: ordinal conditioning based on the minimum operator which is suitable for the ordinal setting:

$$\pi(w \mid_m B) = \begin{cases} 1 & \text{if } \pi(w) = \Pi(B), w \in B \\ \pi(w) & \text{if } \pi(w) < \Pi(B), w \in B \\ 0 & otherwise \end{cases} \tag{1.9}$$

- Quantitative setting: Conditioning based on the product operator and suitable for the numerical setting:

$$\pi(w \mid_p B) = \begin{cases} \frac{\pi(w)}{\Pi(B)} & \text{if } w \in B \\ 0 & otherwise \end{cases} \tag{1.10}$$

Quantitative possibilistic conditioning can be viewed as a particular case of Dempster's rule of conditioning since possibility measures are special cases of plausibility functions [156].

Possibility theory have many definitions of *independence* [5]. In particular, two common definitions have been used to develop possibilistic networks as will be presented in the next section:

- Non interactive relation [174]: This relation is based on the ordinal conditioning and is defined as follows:

$$\Pi(x \wedge y \mid z) = \min(\Pi(x \mid z), \Pi(y \mid z)), \forall x, y, z. \tag{1.11}$$

- Product based independence relation: this relation is based on product based conditioning and is defined by:

$$\Pi(x \wedge y \mid z) = \Pi(x \mid z).\Pi(y \mid z), \forall x, y, z. \tag{1.12}$$

1.4.4 Possibilistic networks: PN

Existing works on possibilistic networks are either a direct adaptation of the probabilistic approach [8] or a way to perform learning from imprecise training data [16], or from imprecise training and testing data [93].

Possibility theory provides two ways to define a counterpart of Bayesian networks (presented in Chapter 2) depending on if we use a ordinal or a product based possibilistic conditioning. Product-based possibilistic networks are very similar to probabilitic based networks. Minimum based possibilistic networks differ from them. The key difference concerns the recovering of the initial data from the network, which is not ensured in minimum based networks.

Possibilistic networks Representation

A directed possibilistic network on a variable set V is characterized by a graphical component and a numeric component.

- **A graphical component:** It is a *Directed Acyclic Graph* (DAG). The graph structure encodes independence relation sets between nodes. Each node in the graph represents a domain variable and each link represents a dependency between two variables. The DAG enables representing conditional dependency between dependent or independent variables.

- **A numerical component:** it quantifies distinct links in the graph and represents conditional possibility matrix of each node in the context of its parents. These possibility distributions should respect normalization.

For each variable V:

• If V is a root node and $Dom(V)$ the domain of V, the prior possibility of V should satisfy:

$$\max_{v \in Dom(V)} \Pi(v) = 1 \tag{1.13}$$

• If V is not a root node, conditional distribution of V in the context of its parents denoted U_V should satisfy:

$$\max_{v \in Dom(V)} \Pi(v \mid u_V) = 1, \forall u_V \in Dom(U_V) \tag{1.14}$$

where U_V is the value of parents of V and $Dom(U_V)$ is the domain of parent set of V .

Given all conditional and priori distributions, joint distribution relative to all variable set can be expressed by the following rule chain:

$$\pi(V_1, ..., V_N) = \otimes_{i=1..N} \Pi(V_i \mid U_{V_i}) \tag{1.15}$$

Where $\otimes$ is a t-norm operator which could be the *min* or the *product*. Possibilistic networks are too related to the adopted conditioning type, it is called a Min-based Possibilistic Network when we use a Min-based conditioning and a product-based Possibilistic Network when we rather use a product-based conditioning.

Product-based Possibilistic Network A product-based possibilistic graph, denoted by PPG, is a possibilistic graph whose associated conditional possibility distribution is based on the product operator. The PPG is mainly suitable for the numerical setting where possibility measures represents numerical values in [0, 1].

The possibility distribution of product-based possibilistic networks π_p, obtained by the associated chain class is:

$$\pi(V_1, ..., V_N) = \prod_{i=1}^{N} \Pi(V_i \mid U_{V_i}) \tag{1.16}$$

Min-based Possibilistic Network: A Min-based possibilistic graph, denoted by MPG, is a possibilistic graph based on the minimum operator suitable for the ordinal setting where only order between possibility measures is taken into account. The possibility distribution of Min-based possibilistic networks can be derived from the following:

$$\pi(V_1, ..., V_N) = \min_{i=1}^{N} \Pi(V_i \mid U_{V_i}) \tag{1.17}$$

Example 1.2.

The following figure represents an example of a possibilistic network:

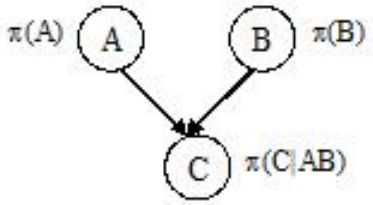

Figure 1.4: An example of a Possibilistic Network

Table 1.1 shows priori and conditional possibility distributions corresponding to variables A, B, and C. Using the chain rule in (equation 1.16) based on the product-operator, Table 1.2 includes the joint distribution for the product based setting. In the same manner, we can obtain posterior possibility distributions in the min based setting, in particular: $\pi(a_2b_1c_2) = min(\pi(a_2), \pi(b_1), \pi(c_2|a_2b_1)) = min(0.7, 0.3, 0.2) = 0.2$

Table 1.1: Initial distributions

a	$\Pi(a)$	b	$\Pi(b)$	a	b	c	$\Pi(c \mid a \wedge b)$	a	b	c	$\Pi(c \mid a \wedge b)$
a_1	1	b_1	0.3	a_1	b_1	c_1	0.5	a_2	b_1	c_1	1
a_2	0.7	b_2	1	a_1	b_1	c_2	0.3	a_2	b_1	c_2	0.2
				a_1	b_2	c_1	1	a_2	b_2	c_1	0.8
				a_1	b_2	c_2	0.5	a_2	b_2	c_2	1

Table 1.2: Posteriori possibility distributions

w	$\pi(w)$	w	$\pi(w)$
$a_1b_1c_1$	0.15	$a_2b_1c_1$	0.21
$a_1b_1c_2$	0.09	$a_2b_1c_2$	0.042
$a_1b_2c_1$	1	$a_2b_2c_1$	0.56
$a_1b_2c_2$	0.5	$a_2b_2c_2$	0.7

Propagation in Possibilistic Networks

The main interest of possibilistic networks is the evaluation of the realization impact of certain event on the remaining variables. This process is usually named propagation which enables to compute posterior possibility distributions for each variable given the evidence on the rest of variables.

Many possibilistic propagation algorithms have been proposed in the literature and are especially a direct adaptation of exact or approximate methods proposed for Bayesian networks [83] [17] and have the same NP-complete aspect. In particular, these are some related works that treated propagation in possibilistic networks: [83], [17] and [4].

1.4.5 Information fusion in possibility theory

The need to fusion uncertain pieces of information from different sources is an important issue in many areas, such as, multi-sensor data fusion, image data fusion, expert opinion fusion, and multiple classifier results combination.

The information fusion problem differs from both two other problems:

i) Information revision problem: this problem concerns updating available information upon the arrival of a new piece of information using conditioning as previously described. Information revision has non symmetrical view to knowledge

which assumes that prior knowledge is less evaluated than the new knowledge. However in the fusion problem, all information are treated in a *symmetric* way and sources play the same role even if information is heterogeneous.

ii) Multiple criteria agent preferences: the aim in this problem is to search for a *compromise* between different view of agents whereas in information fusion, the aim is to search for a common knowledge by merging different uncertain information coming from agents.

Uncertainty theories investigate different information merging rules. In particular, probability theory makes use of the Bayesian probabilistic fusion [73], the Dempster-Shafer evidence theory makes use of Dempster's combination rules and possibility theory exploits the disjunctive or conjunctive (or more generally, t-norm and t-conorm) fusion operators. In the following, we will only be interested to develop possibility theory merging rules useful in our searching field in this work.

The choice of the information merging rule is too related to the problem under study and the reliability of information sources [64]. In particular, the conjunctive fusion operator is commonly applied to a set of *reliable* sources which agree with each other. When reliability condition could not totally be checked, the disjunctive fusion operator seems to be more appropriate. Other refining rules, combining conjunctive and disjunctive operators, have been proposed for situation where sources may partially be in agreement and/or only some sources are reliable.

Aggregation rules:

Before going through the development of information merging rules, we first give a formal representation of the fusion problem.

Given two information sources S_1 and S_2, the fusion problem account to search for a value of a variable w considering that:

1. S_1 affirms that $w \in A_1$
2. S_2 affirms that $w \in A_2$
3. $A_1, A_2 \in \Omega$

Information provided by each source is usually a possibility distribution π defined on Ω which reflects the expert belief about the value of the variable w. If n sources are to be considered, the problem amounts to merge these pieces of information $(\pi_i, i = 1, ..., n)$ in order to define a new possibility distribution π which is a function of all other possibility distributions coming from different sources. More formally:

$$\pi(w) = f(\pi_1(w), ..., \pi_n(w)), \forall w \in \Omega \tag{1.18}$$

A complete overview of aggregation rules can be found in Dubois and Prade [62], Fodor and Yager [82].

Conjunctive fusion Triangular norms ($t-norms$) [110] are mainly used for conjunctive fusion. Such functions are associative, monotonically increasing, and have a neutral element 1. The conjunctive fusion rule is defined by:

$$\pi(w) = \otimes_{i=1..n}\pi_i(w), \forall w \in \Omega \tag{1.19}$$

Conjunctive fusion is mainly suitable for problems where all information sources are reliable and each of them agree with each other. This combination mode uses intersection between pieces of information to derive the resulting conclusion. In particular, there are the *minimum*, the *product*, and the *linear conjunction* rule:

- *The minimum based-rule* : $a \otimes b = min(a, b)$
- *The product based-rule* : $a \otimes b = a.b$
- *The linear conjunction rule* : $a \otimes b = max(0, a + b - 1)$

Given *a, b, c* and $d \in [0, 1]$, the following axioms can be checked for t-norms.

1. **Boundary conditions:** $a \otimes 1 = a$ and $a \otimes 0 = 0$
2. **Monotonicity:** $a \otimes b \leq c \otimes d$ if and only if $a \leq c$ and $b \leq d$
3. **Commutativity :** $a \otimes b = b \otimes a$
4. **Associativity :** $(a \otimes (b \otimes c)) = ((a \otimes b) \otimes c)$.

When using $t-norms$, the main question is which rule could be more appropriate to a given situation?

To answer to this question, we should study characteristics of each combination operator. In particular, the *min* based rule enables preserving idempotency which means that if the n sources give the same possibility distribution it is this distribution which will be considered in result. This case is faithful to avoid redundant information. The *product* based rule is assumed to be more applicable than the *min* when sources are assumed to be independent.

This *product* operator have a more reinforcement effect on the resulting distribution than the *min* in the sense that the combined information will usually have a possibility degree very small if compared to possibility degrees assigned by sources. Finally the *linear conjunction* rule have a stronger reinforcement effect since it discards any information that is considered as very little plausible by all sources.

In more recent work, Liu et al. [123] addressed the problem of selecting the best merging rule. The conjunctive fusion usually generates sub-normalized possibility distributions (i.e. $\pi(w) < 1, \forall w \in \Omega$) mainly when sources are very inconsistent and do not agree that there is one totally plausible value for a variable w.

This result could be normalized as follows:

$$\pi(w) = \frac{\otimes_{i=1..n}\pi_i(w)}{max_{w\in\Omega} \otimes_{i=1..n} \pi_i(w)}, \forall w \in \Omega \tag{1.20}$$

The associativity is preserved when the product operator is used in this normalization and is lost when the minimum is applied.

Disjunctive fusion *Triangular conorms* ($t-conorms$) [110] are dual to $t-norms$ and are the principal used disjunctive fusion. The disjunctive fusion rule is defined by:

$$\pi(w) = \oplus_{i=1..n}\pi_i(w), \forall w \in \Omega \tag{1.21}$$

Disjunctive fusion is applied when information sources disagree and when we are sure that one of the sources is reliable but, given the available knowledge about these sources, we are enable to know which one is the reliable.

This combination mode assumes that if propositions are contradictory, then it is better to consider the maximum consistent subsets of propositions by assuming that the real proposition is one of them. This logic comes from the fact that if propositions are very discarded (intersection gives the empty set with t-norms very close to 0), it is more natural to say that one of them may be reliable rather than to say that simply we are in a total conflict and no proposition could be assumed. The following disjunctive fusion rules are usually used:

- *The maximum based-rule* : $a \oplus b = max(a, b)$
- *The probabilistic sum* : $a \oplus b = a + b - a.b$
- *Bounded sum* : $a \oplus b = min(1, a + b)$

Given a, b, c and $d \in [0, 1]$, the following axioms can be checked for $t-conorms$.

1. **Boundary conditions:** $a \oplus 1 = 1$ and $a \oplus 0 = a$
2. **Monotonicity:** $a \oplus b \leq c \oplus d$ if and only if $a \leq c$ and $b \leq d$
3. **Commutativity :** $a \oplus b = b \oplus a$
4. **Associativity :** $(a \oplus (b \oplus c)) = ((a \oplus b) \oplus c)$

Contrary to conjunction fusion, this disjunctive fusion provides normalized possibility distributions. Nevertheless, the main disadvantage of disjunctive merging operator is that the obtained distribution could be very imprecise since this operator assumes that only one source is reliable. In particular when possibility distributions given by sources is very inconsistent, we risk to obtain a possibility equal to 1 for all subsets, which represents total ignorance.

More refined fusion rules have been proposed as an attempt to integrate both conjunctive and disjunctive operators into a single merging rule. These rules are basically more suitable for problems where only some sources are reliable and that they partially agree with each other. We cite for example the weighted fusion operator, the adaptive fusion and the accumulative fusion. More details about these fusion modes is in [64].

1.4.6 Possibilistic logic

In this section, we briefly restate the background on standard possibilistic logic. For a complete presentation of possibilistic logic you can see [57].

Possibilistic logic, developed by [57] [72] is a convenient tool for handling uncertain or prioritized pieces of information. Possibilistic logic includes a set of logical formulas associated with a certainty factor (weights) and an inference process used in deductive reasoning.

Before going for more details, we will first give principal notations used in this section:

- Let $\phi, \varphi, \chi, ...$ denote logical formulas.
- $\alpha, \beta, ...$ denote certainty factors or necessity measures associated to formulas.
- let $\neg$, $\wedge$ and $\vee$ are respectively the negation, the conjunctive and the disjunctive operators.
- Let $w \in \Omega$ be a possible interpretation.
- $\models$ denotes a syntactic inference.

- $\perp$ and $\top$ are respectively contradiction and tautologies.

A possibilistic logic expression is a pair (ϕ, α), where ϕ is a first-order formula and $\alpha \in (0, 1]$ corresponds to a lower bound of a necessity measure N (also called *certainty factor*). A formula of the form (ϕ, α) means that ϕ is certain at least to the degree α ($N(\phi) \geq \alpha$), where N is a necessity measure. In the qualitative framework, an ordered scale can be used instead of the numerical [0, 1] scale.

Possibilistic Knowledge base

A possibilistic knowledge base denoted K is a set of logic formulas defined as follows:

$$K = \{(\phi_i, \alpha_i), i = 1, ..., n\}$$

n is the number of formulas in K, usually formulas are presented as conjunctions associated to their necessity measure. We note that the knowledge base K covers a classical knowledge base where all propositions have the same certainty factor 1.

A possibilistic knowledge base is characterized by a possibility distribution π_K representing a fuzzy set on interpretations $w \in \Omega$ of K. A distribution π_K on Ω is a function from Ω to [0,1]. π_K is said to be normalized if $\exists w \in \Omega$ such that $\pi_K(w) = 1$. A normalized distribution codes a total order on interpretations defined as follows:

$$\pi_K(w) = \begin{cases} 1 & \text{if } w \models \phi_i, \forall(\phi_i, \alpha_i) \\ (1 - \alpha_i) & otherwise \end{cases}$$

A key issue in possibilistic logic is to define and treat inconsistent knowledge bases. Inconsistency is defined by the highest necessity of the interpretation in K which leads to a contradiction:

$$Inc(K) = max(\{\alpha; K \models (\perp, \alpha)\}.$$

K is also called a knowledge base of α level of inconsistency, which means that all proposition $\phi_i \in K$ has certainty factor α_i greater or at least equal to α.

Necessity and possibility measures

A necessity measure N is a function defined on a set of logical formulas to a totally ordered bounded set with the following axioms:

1. $N(\top) = 1$,
2. $N(\bot) = 0$,
3. $N(\phi \wedge \varphi)) = min(N(\phi), N(\varphi))$

The last axiom states that to be certain about $\phi \wedge \varphi$ to a certain degree, we should be certain (at least) as much as ϕ and φ separately.

The possibility degree of each interpretation w can be estimated by the dual measure of the maximum necessity of a formulas satisfied by w. This means that, w is impossible ($\pi(w) = 0$) if there exists a formula with a certainty factor 1 falsified by w. From the possibility distribution π_K, the possibility measure is defined for formula ϕ which is a dual measure of necessity N:

$$\Pi(\phi) = 1 - N(\neg\phi) = max\{\pi_K(w), w \in \Omega, w \models \phi\}.$$

Possiblistic inference

The min-decomposability of necessity measures enables to work with weighted clauses without lack of generality, since $N(\wedge_{i=1..n}\phi_i) \geq \alpha \iff \forall i N(\phi_i) \geq \alpha$. This means that a weighted conjunctive logical formula is equivalent to a set of weighted clauses as follows:

$$(\wedge_{i=1..n}\phi_i, \alpha) \equiv \wedge_{i=1..n}(\phi_i, \alpha).$$

The basic inference rules in possibilistic logic are [57] :

- **Resolution:** $(\neg\phi \vee \psi, \alpha); (\phi \vee \varphi, \beta) \models (\psi \vee \varphi, min(\alpha, \beta))$
- **The weight weakening:** for $\beta \leq \alpha, (\phi, \alpha) \models (\phi, \beta)$

Note that classical inference is retrieved when all the weights are equal to 1. The following inference rules are also valid for propositional formulas:

- **Formula weakening :** if ϕ entails ψ classically, $(\phi, \alpha) \models (\psi, \alpha)$.
- **Weight fusion:** $(\phi, \alpha); (\phi, \beta) \models (\phi, max(\alpha, \beta))$.

Refutation can be easily extended to possibilistic logic. Given the knowledge base K, proving (ϕ, α) from K returns to add $(\neg\phi, 1)$ in a clausal form to K and using the previous rules show that $K \cup (\neg\phi, 1) \models (\bot, \alpha)$.

A complete presentation of resolution and refutation based inference in possibilistic logic is given in [57].

1.5 Links between probability and possibility theories

A variety of works are interested, in the literature, to compare the existing uncertainty theories. Beyond giving only some differences which may be uninterpreted, it is clearly difficult to conclude the superiority of one of the previously presented theories. It was proven in [73][60] that, if mathematical object are close, the belief function theory can be viewed as more general than probability and possibility theories since they can be considered as a particular case, in fact:

- If the mass is attributed to only singleton, the mass is called Bayesian, the construction of the corresponding belief function gives only one probability measure $bel = P = pl$.
- If the mass is attributed to focal element A_i which are consonant ($i < j \Rightarrow A_i \subseteq A_j$), then the corresponding belief function is a necessity function ($bel = N$) and the plausibly function is a possibility measure ($pl = \Pi$).

However combination operators are different and the result of the combination of two consonant belief masses is rarely consonant. Moreover, we note that the classical statistical inference based on likelihood (which is not a probability) and if normalized, is similar to a possibility distribution.

These are some reflections which incites us not to consider these theories as rivals, but as proposing complementary representations of uncertainty. A more deep comparison between these theories is necessary to show which theory is more convenient to a special field. In particular, we note the interest of handling conjointly different formalisms notably to manipulate heterogeneous data. For this reasons some authors have proposed transformations enabling to pass from one formalism to another [61][34][48][111][119] [59].

In the following, we only give further reflections on comparison between probability and possibility theories and then present a widely used transformation method from probability to possibility and inversely in which we are only concentrated on the numerical case.

A main difference between a possibility distribution π and a probability function p is that the latter is a normalized measure that requires that the probability sum of elements in the universe of discourse is equal to 1, whereas in possibility theory no constraint of this type is required. Probability theory claims that the probability of an event could simply determined by the probability of the complement of that event whereas possibility theory involves non-additive measures. So, the crucial disadvantage of the use of probabilities in uncertainty representation resides in the

necessity to list an *exhaustive* set of *mutually exclusive* alternatives. In real world problems, it is difficult for an expert to provide facts that are exhaustive and mutually exclusive because his/her knowledge increases along time and so uncertainty about the situation decreases.

In the other hand, possibility theory could represent an ordinal scale of uncertainty where only the order between events is important to consider. In some situation, an expert could only assigns a general order on variables which reflects his/her qualitative belief on their existence. This field of interest is known under the noun *qualitative possibility* which looks similar to *subjective probability* proposed as a counterpart to the quantitative probability based on frequency of events. However the two concepts are radically different in their structure.

It is important to note that the frequency of observations usually provides valuable information. However, the frequency-based approach are usually depending on statistical assumptions when estimating probability distributions. Thus, these later might be misleading if these assumptions are strongly violated.

Moreover, a possibility distribution is more expressive in some situations. In fact, possibility measures can reflect *ignorance*: in particular, the distribution $\pi(w) = 1, \forall w \in \Omega$ is an expression of *total ignorance* and reflects the absence of any relevant information, which means that the expert "knows that it doesn't know". We can see, in the following, that this type of uncertainty cannot be clearly represented by the use of probabilities.

Let us continue with the *Constituent Assembly election* example given in Section 1.2. Considering candidate Parties P_1, P_2 and P_3, we assume that $\pi(P_1) = 1$, $\pi(P_2) = 1$ *and* $\pi(P_3) = 1$ which reflects the idea that the three Parties are entirely possible to gain election. If one wants to use probabilities, he could assign the weight 1/3 to each Party. Let us suppose now that a new candidate Party P_4 has been added to the list of politic Parties, but that old Parties keep the same statute as previously presented i.e. that there isn't any supplementary information that comes to reinforce or decrease probabilities for the three first Parties. The possibilities are consequently: $\pi(P_1) = 1$, $\pi(P_2) = 1$, $\pi(P_3) = 1$ and $\pi(P_4) = 1$. However, the use of probabilities would now results in assigning the weight 1/4 to each Party. This is due to the principle of insufficient reason which models complete ignorance by the uniform distribution. This new assignment means that the first three Parties became less "probable" although we have any supplementary knowledge which confirms this redistribution.

Thus we can say that the probability theory can represent *ambiguity* whereas the possibility theory can distinguish between both problems, *ambiguity* and *ignorance*. Ambiguity is present if there is several plausible events with close confidence support and ignorance can be reflected by the fact that even the most supported event has

a small degree of possibility.

An other disadvantage of probabilistic measures has a more practical nature. The calculation rules in probability theory are principally based on products and sums. The use of these operators encourage the propagation of errors essentially when the calculus process is too long. Also, this makes the probabilistic model distinguish much between alternatives whereas this distinction may be unmeaningness in some situations. This can be prevented by using the *min* and the *max* operators which makes the principal sense of ordinal possibility theory.

Finally as presented in Section 1.4.2, a possibility degree can be viewed as, an upper bound of probability degrees [66]. Let $\wp_\pi = \{P, \forall A, N(A) \leq P(A) \leq \Pi(A)\}$ be the set of probability measures encoded by a possibility distribution π. This notation is coherent with imprecise probabilities since $P^*(A)$ and $P_*(A)$ are respectively Π and N.

In the following we only present transformation from probability to possibility and inversely which will be basically used in next chapters. Further transformations between other formalisms such us belief function and probability or possibility could be found in [73].

1.5.1 Probability to possibility transformation

There are several transformations for moving from the probability framework to the possibility framework based on various principles such as consistency (what is probable is possible) or information invariance [35, 47, 111, 67, 58].

The transformation from probability to possibility distributions [58], which has been extended to continuous universes, accounts for epistemic uncertainty. It yields the most restrictive possibility distribution that is co-monotone with the probability distribution and that provides an upper-bound on the probability of any event.

Let us recall the principle underline the transformation from probability distribution p to possibility distribution π^* proposed in [58][74]. The resulting possibility distribution should satisfy the following properties:

- Possibility - probability consistency: For any probability density p, the possibility distribution π^* is consistent with p, that is: $\forall A, \Pi^*(A) \geq P(A)$, with Π^* and P being the possibility and probability measures associated to π^* and p respectively.

- Co-monotony of distributions: $\pi(w) > \pi(w')$ if and only if $p(w) > p(w')$.

The rationale behind this transformation is that given a probability p, one tries to preserve as much information as possible. This leads to select the most specific

element in the set $PI(P) = \Pi : \Pi \geq P$ of possibility measures dominating P such that $\pi(w) > \pi(w')$ iff $p(w) > p(w')$.

Dubois et al.[74] suggest to use the "maximum specificity" principle which aims at finding the most informative possibility distribution that encodes the considered probability information. A possibility distribution π_1 is more specific than a possibility distribution π_2 if and only if

$$\forall x \in \Omega, \pi_1(x) \leq \pi_2(x).$$

Since a possibility distribution explicitly handles the imprecision and is also based on an ordinal structure rather than an additive one, it has a weaker representation power than a probability one. This kind of transformation (probability to possibility) may be desirable when we are in presence of poor source of information, or when it is computationally harder to work with the probability measure than with the possibility measure.

In the case where the universe of discourse is discrete (i.e. $\Omega = \{w_1, \ldots, w_n\}$), the most specific possibility distribution π^* given a probability distribution p over Ω is defined by:

$$\forall i \in \{1, \ldots, n\}, \pi^*(w_i) = \sum_{w_j | p(w_j) \leq p(w_i)}^{n} p(w_j). \tag{1.22}$$

Example 1.3.

If we consider $\Omega = \{w_1, w_2, w_3\}$ and p such that $p(w_1) = 0.5$, $p(w_2) = 0.3$ and $p(w_3) = 0.2$. We obtain $\pi^*(w_1) = 0.5 + 0.3 + 0.2 = 1$, $\pi^*(w_2) = 0.3 + 0.2 = 0.5$ and $\pi^*(w_3) = 0.2$.

Dubois et al. [74] have justified a probability-possibility transformation method in the *continuous* case in terms of confidence intervals (with level ranging from 0 to 1) built around a nominal value which is the mode. It generalizes the previously presented method for the discrete case [74]. In this context, densities are assumed to be symmetric with unique mode. Then, the mode is equal to the mean and to the median. A confidence interval I_α represents the smallest range of values that is believed to include the "true" value of the considered variable, with a fixed probability α. Its confidence level is $P(I_\alpha) = \alpha$ (usually 95%), $1 - P(I_\alpha)$ is the risk level, that is, the probability for the real value to be outside the interval. It leads to build the following possibility distribution Π^* in the continuous case:

$$\pi^*(x) = sup\{1 - P(I^*_\alpha), x \in I^*_\alpha\}. \tag{1.23}$$

Where I_α is the $\alpha\%$ confidence interval.

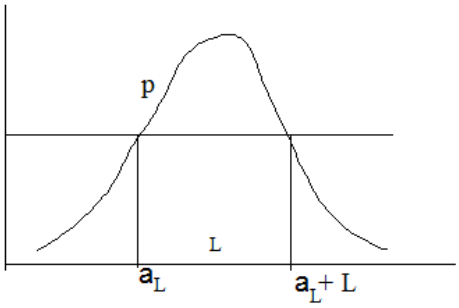

Figure 1.5: Confidence Interval I_L for a given a_L

To model this possibility distribution, the authors replace a unimodal probability distribution by a collection of intervals I_L with their confidence level α such that:

$$\pi(w) = \begin{cases} 1 & \text{if } w \in I_L \\ (1-\alpha) & \text{if } w \notin I_L \end{cases}$$

Where I_L is the smallest interval of the collection that contains w. The most specific possibility distribution thus obtained satisfies the above requirements: consistency with p, and comonotony:

$$\forall L > 0, \pi(a_L) = \pi(a_L + L) = 1 - P(I_L). \tag{1.24}$$

where I_L is the smallest confidence interval, of length L, that contains a_L (see Figure 1.5).

Figure 1.6 presents the maximally specific probability-possibility transformation (in blue) of a normal distribution (in green).

1.5.2 Possibility to probability transformation

Dubois et al. [74] proposed a possibility to probability transformation which is based on the Laplace principal. This later claims that anything equiplausible should be equiprobable.

Given a possibility distribution π, the problem is to find a probability distribution p satisfying the following properties:

$$\forall A, P(A) \leq \Pi(A) \text{ (probability-possibility consistency)}$$

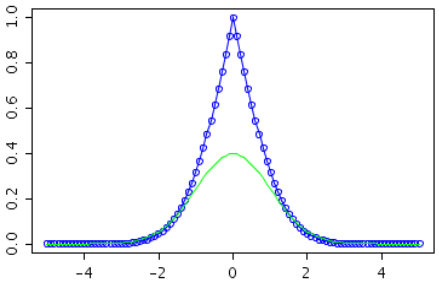

Figure 1.6: The maximum specific possibility distribution for $\mathcal{N}(0,1)$.

$$\pi(w) > \pi(w') \Leftrightarrow p(w) > p(w') \text{ (preference preservation)}$$

$$p \text{ contains as much uncertainty as possible}$$

The authors use the *insufficient reason principle* to satisfy the last property. To apply this principle, one can assume that the maximum uncertainty about w can be represented by a uniform probability distribution on A.

In the discrete case, given a possibility distribution π which can be described by a finite set $w_1, w_2, ..., w_n$ corresponding to $\pi(w_1) = 1 > \pi(w_2) > ... > \pi(w_n) > \pi(w_{n+1}) = 0$, the following density p is the transform of π:

$$p(w_i) = \sum_{j=i}^{n} \frac{\pi(w_j) - \pi(w_{j+1})}{j}, \forall i = 1..n \tag{1.25}$$

Example 1.4.

Let $\Omega = \{w_1, w_2, w_3, w_4\}$ and a possibility distribution π such that $\pi(w_1) = 1$, $\pi(w_2) = 0.5$, $\pi(w_3) = 0.3$ and $p(w_4) = 0$. We obtain $p^*(w_1) = (1 - 0.5)/1 + (0.5 - 0.3)/2 + (0.3 - 0)/3 = 0.7$, $\pi^*(w_2) = (0.5 - 0.3)/2 + (0.3 - 0)/3 = 0.2$, $\pi^*(w_3) = 0.1$ and $\pi^*(w_4) = 0$.

In the continuous case, given the possibility distribution π, the main idea is to estimate a probability distribution from $\alpha - cuts$ of the possibility distribution π, the transformation is:

$$\forall w \in \Omega, p(w) = \int_0^{\pi(w)} \frac{d\alpha}{\mid (\pi)_\alpha \mid} \tag{1.26}$$

where $(\pi)_\alpha = \{w, \pi(w) \geq \alpha\}$ is the α-cut of π, and p is a probability density.

If we consider the case where π is a membership function with a support $[S_L, S_U]$ and a core $[C_L, C_U]$ defined by:

$$\pi(w) = \begin{cases} F(w) & \text{if } w \in [S_L, C_L] \\ 1 & \text{if } w \in [C_L, C_U] \\ G(w) & \text{if } w \in [C_U, S_U] \end{cases}$$

This transformation become:

$$p(w) = \int_0^{\pi(w)} \frac{d\alpha}{(G^{-1}(\alpha) - F^{-1}(\alpha))} \tag{1.27}$$

If π is a trapezoid function such that:

$$F(w) = (w - S_L)/(C_L - S_L), F^{-1}(\alpha) = (C_L - S_L)\alpha + S_L$$
$$G(w) = (w - S_U)/(C_U - S_U), G^{-1}(\alpha) = (C_U - S_U)\alpha + S_U$$

we obtain:

$$p(w) = \int_0^{\pi(w)} \frac{d\alpha}{(p\alpha + q)} \tag{1.28}$$

$$p(w) = 1/p.Ln((p.\pi(w) + q)/q) \tag{1.29}$$

where: $p = C_U - S_U - C_L + S_L$ and $q = S_L - S_U$.

Transformation of other kinds of possibility distributions can be found in [74]. For a more deep analysis of the two uncertainty paradigms you can see [32]. In this work, the authors study the coherence between the probability and possibility measures obtained using possibility distribution and density functions in the continuous case.

1.6 Conclusion

This chapter offers a summary of imperfection in data representation where a distinction is made between different forms of imperfect data in particular imprecision and uncertainty.

In the second part, we have briefly recall basic notions of theories dealing with uncertainty, principally probability theory, belief function theory and fuzzy set theory. In particular, possibility theory is a new tractable theory that offers a common setting for modelling imprecision and uncertainty.

In a large part of this chapter, we investigate an overview of basic concepts related to possibility theory mainly possibilistic distributions, possibilistic conditioning, possibilistic networks, fusion and possibilistic logic, etc. In the last part, we involve a discussion between uncertainty theories and we have mainly emphasized comparison between probability and possibility theories. Probability-possibility transformation methods are also presented in the discrete and continuous cases.

In the next chapter, we will focus on the most important machine learning techniques, namely classification. Our interest to possibility theory comes from the intuition that this theory, if merged with an appropriate classification method, could be a promising tool for classifying imperfect data.

Chapter 2

Classification methods

2.1 Introduction

Low costs of machines in term of storage and power computing and also the development of data-processing techniques encouraged the companies and organizations to accumulate large masses of data. These databases are usually under exploited in spite of they can contain strategic knowledge that can help experts in their reasoning.

These data reserves constitute an important information mine which companies must exploit and explore in order to discover relevant information useful for prediction and decision making. Data Mining (DM) or Knowledge Discovery in Databases (KDD) are convenient tools for data exploring need and they include many research domains principally: databases, statistics and machine learning.

Actually Machine Learning (ML) is the most popular, in this field, which concerns the searching and discovering of relation, links and logic rules from instance bases. Machine learning is the origin of a large number of algorithms used for knowledge discovery and applied in several domains such as medicine, physics and industry.

These algorithms vary in the context of the solution they propose, the data they exploit, the model they generate and the strategy for training. In spite of these differences, all ML techniques seek to find an acceptable generalization of the problem when learning from possible space of concepts. The learning problem amount for double objectives: avoiding the combinative explosion due to the vastness search field of possible generalizations and selecting good generalization from all possible ones. There are several learning methods which are principally divided on supervised, unsupervised or semi-supervised learning.

Supervised learning concerns all algorithms that exploit labeled training data, where the desired outputs are usually provided by human experts, to generate a

general model that maps inputs to desired outputs. In unsupervised learning, the aim is to analyze the general organization of data where the learner is given only unlabeled instances. The semi-supervised learning generates learner which combines both labeled and unlabeled instances.

Among this panoply of machine learning techniques, we emphasize $classification$ problems as a supervised learning which aims to approximate a model or a function mapping input vectors into output classes by looking at training input-output examples.

In this chapter we will mainly be interested to classification task and we will present an overview of the most used classification techniques. Especially, two families of methods are described: methods based on risk minimization such as decision trees and artificial neural networks and methods based on the maximum likelihood estimation such as Bayesian networks and instance-based learning.

For these techniques, we only give the basic principle and we principally focus more on Bayesian networks which are detailed in this chapter because they constitute the background for possibilistic classification approaches that we propose in this work. This chapter also includes a comparative study of these classification techniques which discusses their positive and negative aspects.

This chapter includes seven sections: the first five sections corresponds respectively to a brief recall of decision trees, rule-based learning, neural networks, support vector machine and instance-based learning techniques. Section 6 presents the Bayesian networks and the main existing algorithms of Naive Bayesian Classifiers. In Section 7, we give a comparative study between different techniques.

2.2 Notations for classification techniques

In the following, we give some notations to unify the learning process for all classification techniques:

- $A = \{A_1, A_2, ..., A_M\}$: the set of M **attributes** (properties) or domain variables of objects in the problem under study.
- $V(A) = \{a_1, a_2, ..., a_M\}$: the **value** of each numerical attribute in A which can be a real number $\Re$ or an interval in the numerical case.
- $C = \{c_1, c_2, ..., c_C\}$: the set of C discrete labels or **classes**, i.e., the output variable for each object.

- $T = \{a, b, ...\}$: the set of all examples or *instances* corresponding to all observed examples in the universe of discourse. More formally each a_k is represented by $V(A)$.
- $T_r = \{a_{tr}, b_{tr}, ...\}$: the **training set** (a subset of T) containing N labeled/unlabelled instances used in the learning phase. In supervised learning, an instance include both $V(A)$ combined with a class value c_j whereas in unsupervised learning only $V(A)$ is given in each instance.
- $T_s = \{a_{ts}, b_{ts}, ...\}$: the **testing set** that contains all instances in T not exploited for training. This set is used to test the generalization ability of the classifier.

2.3 Decision Trees

For many learning tasks, it is important to produce classifiers that are not only highly accurate, but also easily understood by humans. Decision Trees (DTs) are a particular type of rule-based tools. The attractiveness of these models rests on the rules they learn which can simply be understandable and interpretable. Supervised learning algorithms uses recursive partitioning to form a tree structure with if-then rules (each of them is applied with a particular feature as splitting criteria). These tests are chosen to best discriminate among target choices. Each branch on different levels of the tree represents a subgroup of observations with homogeneity of different degrees. Homogeneity increases from top to bottom where the bottom leaves contain the cases with the same target (classes) while the top branches offer the roughest split.

Decision trees are generated in two phases:

i) The building phase: two main steps are conducted here. The first one concerns the tree structure growing and the second one consists in optimizing by punning the tree induced in the first step.

ii) The classification phase: the induced decision tree is used to classify test instances.

To induce a decision tree, a training set is usually used which can include either discrete or continuous attributes.

For a given training set, the general idea when constructing a DT, is to create a test on an attribute in each construction step. Then the main question is how to choose the best attribute to create branches in the tree. Several methods can be applied to define the optimal test, however the objective is always the same : choose

the test on an attribute that best discriminates classes in the training data. This process is repeated on each partition of the divided data, creating branches in the tree until training data in divided on subsets labeled with the same class. At this moment, a leaf node is created and the target class is assigned to this leaf.

The most used methods for feature selection is the $Information\ Gain$ [100] and the *Gini Index* [27]. In particular, the information gain computes the estimated entropy reduction of examples when a particular attribute is selected for splitting. The DT algorithm calculates this value for each attribute and chooses the one reducing more the entropy, i.e. which contributes to best separate the remaining examples. The general principal of building a decision tree is presented in Algorithm 1.

Algorithm 1 Basic Algorithm for building Decision Tree

if all training examples belong to only one class **then**
  make a leaf node labeled with that class
else
  repeat
    Given a training set T_r, select the attribute that maximizes the splitting criteria such as Information Gain
    Split T_r into different subsets, one for each value of the selected attribute.
  until a stopping criteria is satisfied
end if

In the tree construction phase, an important problem usually occurs; overfitting. The DT classifies very well training examples, but it fails in generalization; it finds difficult to classify unseen examples [134]. Overfitting can occur when examples are noisy or there is'nt enough examples for a particular spitting. In general, the risk of overfitting increases when the tree becomes deep. To ovoid overfitting, the idea is to find the optimal depth of the tree.

To estimate the best size of the tree, we divide the training set in three partitions, then we use only 2/3 for DT construction and the remaining 1/3 for tree validation or punning. The validation procedure consists to successively remove some nodes in the tree. Each removed node becomes a leaf with the same label as its immediate parent node.

Other pruning methods have been proposed, notably those used in the C4.5 algorithm [145]. In this technique, a DT model can simply be presented by a set of rules of the form IF(Premise) THEN (Conclusion). In the pruning step, the algorithm progressively eliminates selectors (conditions) in the premise part. A selector is only removed if it does not decrease the capacity of the tree to classify examples. The final set of rules may contain redundant rules because some examples can be covered by more than one rule. Rules are then ordered: rules making the lowest error rate (False Positive) are presented initially. Examples not covered by any rule are assigned to a default class which is generally the largest class. Pruning reduces the number of

rules and selectors in each rule without affecting the hole accuracy.

The most commonly used DT algorithms include the 1R algorithm which produces the most simple decision rule [96]. This algorithm seeks for the best attribute with many values for which examples of the same class dominates other classes and presents them as a rule. ID3 [146] and its successors C4.5 [145] and C5.0, are actually the most largely used algorithms for DT classification. A study done by Kotsiantis [114] has showed that the C4.5 has the best error rate and rapidity compromise among other machine learning techniques. The principle of this algorithm (and its basic version ID3) will be presented in Chapter 3 as a basic step for inducing a rule-based learner from an initial decision tree.

We also cite the CART algorithm [27] where the abbreviation means "Classification And Regression Trees". This DT algorithm is based on the *Gini index* criteria for attribute splitting and produces only binary branching nodes in the tree. In fact, the C4.5 algorithm is gaining more popularity. Compared to the CART algorithm, this later can produce trees with varying numbers of branches at each node and deal with both continuous and discrete variables

Among machine learning techniques, decision trees are considered as powerful tools which are very useful for prediction. As stated before, they gain their popularity, with respect to other classification techniques, considering their comprehensibility and interpretation ability. Simple rules of the form IF-THEN presented in the DT, are convincing tool able to explain, for a decision maker, why a particular class is assigned to a given instance.

2.4 Rule-based learning

As announced in the previous section, rule induction algorithms, like decision trees, are conceived to acquire general models from training data. These models are presented through a set of classification rules. Each rule is of the form: $IF\ Conj_1\ AND\ Conj_2\ ....THEN\ Conclusion$, where conjunctions describe a selector on an attribute and its value and Conclusion represents predicted value of the class attribute. In this section, we only give a brief introduction to rule-based learning allowing us to compare the general principle of this method to other machine learning techniques. A deep presentation of rule based learners and the corresponding algorithms will be reserved in Chapter 3.

During the 30 last years, several rule induction algorithms have been developed. Almost all these algorithms use one of the following strategies to induce rules from data [134]. The first strategy consists to build a decision tree, using one of the algorithms described in the previous section, then extract one rule for each leaf in

the induced tree. We can cite, in this field, the PART algorithm [85] which is based on the C4.5 algorithm previously introduced. The second strategy, usually based on the *sequential covering* principal, is used in the AQ [132], CN2 [36] and RIPPER [39] algorithms. The third strategy is used in evolutionary algorithms, such as genetic algorithm for rule extraction. We cite for example, the GABIL algorithm [46], and algorithms for genetic programming presented in [86] and [169]. Some other hybrid algorithms combining more than one strategy can also be used.

The sequential covering strategy generates a rule set instead of a decision tree. Rule induction using the sequential covering, learns one rule from the available training set, remove from this set examples that are covered by this rule and then learns recursively an other rule covering the remaining examples. This process is repeated until all examples are covered. This strategy is the most commonly used in a large set of rule induction algorithms. Methods based on this strategy differ principally in the induction procedure used to explore rule candidates and the pruning procedure used to refine induced rules. The rule learning step concerns the feature selection techniques to construct rule selector and the choice of cuts on each feature. This part will be detailed in Chapter 3.

2.5 Neural Network Classifiers

Artificial neural networks are information processing structures containing basic units designed to model the behavior of human neurons. Like the physical architecture of the brain, they are composed of a number of parallel, distributed and interconnected neurons or processing elements (PEs) to produce linear or nonlinear mapping between input and output variables [178]. ANN use pattern association and error correction as an underlying mechanism to represent a problem or relationship. ANN operate as a simple process: a unit or neuron combines its inputs into a single output value, a process usually called the unit's *activation function*. An activation function has two parts: *Combination Function*, which merges all the inputs into a single value; and *Transfer Function*, which transfers the value of the Combination Function to the output of the unit. A network can contain units with different transfer functions serving different operations. Usually the *sigmoid function* is the most used activation function. A common neural network structure used for classification is the topology called multi-layer perceptron (MLP) [151] with an architecture similar to that in Figure 2.1.

In the following we give the general principle of classification with Artificial neural networks. Zhang [175] presents a general overview of the most existing works on ANN. A multilayered perceptron is a network with one or more neurons in the hidden layer (nodes between input and output layers). The number of hidden layer and the number of neurons in each layer are not limited and are too related to the

problem under study. In multilayered perceptron, there is no connection between nodes in the same layer nor loops from output to input nodes.

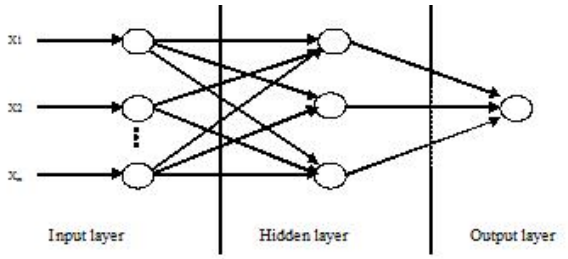

Figure 2.1: A multilayered perceptron

Classification with ANN is conducted into two phases: in the beginning, the network is learned from a training set to determine its structure and then this structure is used to classify new examples.

The network structure defines the relationship between inputs (variables) and outputs (classes). The first learning step enables adjusting activation weights of the network: weights, initially randomly assigned to nodes, are iteratively adjusted. Supervised learning uses labeled examples (pairs of input-outputs) to enable the network to correct its responses in a manner to get closer to the desired responses. In non supervised learning, only input data are given to the network and we expect that this later will be able to define correlation with outputs.

The main problem when learning the structure of the network is the choice of the hidden layer size. In fact, an under-estimation of the node number in this layer may lead to erroneous approximation and so a week generalization ability. In the other hand a high number of redundant units may cause overfitting. Generally, a failure in the application of a NN can be attributed to a inappropriate learning, inadequate number of units in the hidden layer or to the presence of a stochastic instead of a determinist relation between inputs and outputs. A complete study of this problem can be found in [31] in which the authors give a justification of the choice of the network structure.

ANNs are based on the evaluation of the error between the predicted response given by the network and the real response. The network recalculates the connection weights in order to minimize this error. The most known algorithm for supervised learning to estimate weights is the *Back Propagation* algorithm [122] presented in the following. The back propagation is a generalization of the LMS algorithm (Least Mean Square), which uses the gradient descent technique to minimize the error square sum between desired outputs and predicted ones. The back propagation algorithm requires a non linear activation function of neurons which should be differentiable. The most widely used activation function is the *Sigmoid*.

The back propagation algorithm includes the following steps:

- In the first step, the network is initialized with arbitrary weights.
- Present a set of training examples $a, b, ...$ and their desired outputs, $c_1, c_2, ...$ to the network.
- Compute the response $y_1, y_2, ...$ of the network to input examples $a, b, ...$ and their weights.
- Adjust the weights of each neuron to minimize the local error. The weight update is carried out through the following formula:

  $W_{ij}(t+1) = W_{ij}(t) - \varepsilon\ \alpha_{\ i} o_j$ where o_j is the output of the neuron j and ε is the training level.

  - If i is an output neuron: $\alpha_{\ i} = 2(c_i - y_i) f^{'}(I)$, where I is the total input for this neuron ($\sum_{\ j} w_{ij} o_j$).
  - If i is a neuron in the hidden layer : $\alpha_{\ i} = \sum_{\ h} \alpha_{\ h} W_{hi} f^{'}(i)$, where h are neurons connected to i.

These steps are repeated until weights W_{ij} become more steady. The network do not carry out a feedback connection however the error is propagated, in the training phase, from outputs to inputs which justifies the name : "error back propagation algorithm".

The multilayered perceptron has a high ability to generalization i.e. predict responses for examples not observed in the training set. The multilayered perceptron has been used in different domains such as classification, expectation and control. Although the back propagation algorithm allowed to the neuron networks to conquer several real world domains such as pattern recognition, control quality and robotics it has the drawback of being very slow at the training time for the majority of applications. One of the suggested approaches to accelerate training is to initialize the network by optimal weights [170]. The genetic algorithm as well as Bayesian Networks were proposed as alternative to learn NN weights and thus improve its response time.

2.6 Support Vector Machines

The *Support Vector Machine* (SVM) [164], used in the classification task, is based on the principal of structural risk minimization contrary to empirical methods which minimize error caused by training examples such as neuron networks. A survey of

SVM approaches can be found in [30]. The SVM method aims to minimizing the hole error (approximate error and estimated error). In its basic form, the SVM method enables to accomplish binary classification which consists to construct an hyperplane described by weight vectors w and bias terms b as shown in Figure 2.2. This hyperplane separates positive and negative examples with a maximal margin.

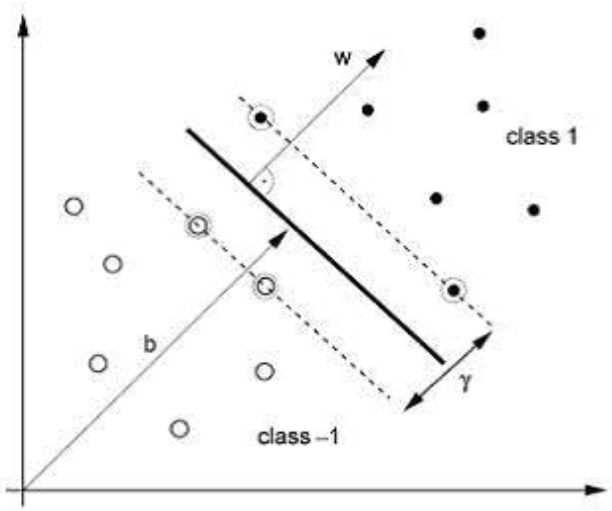

Figure 2.2: Maximum Margin based SVM

Considering the training set with N labeled examples, the SVM method seeks for the optimal hyperplane according to a set of criteria to optimize. In the classification phase, the class to attribute to a test example x, is defined by projecting x on weight vectors w as follow:

$$f(x) = w.x + b \tag{2.1}$$

The sign of the projection enables estimating the predicted class. Several hyperplane choices can be used to separate data space into two subsets. The method based on the maximum margin criteria can be seen as convenient optimality criteria. This criteria favors the hyperplane with the most large separation margin (Figure 2.2). To describe the optimal hyperplane, only vectors on the margin, called *support vectors*, are necessary.

The searching process for this hyperplane returns to an optimization problem which consists at minimizing $1/2\|w\|^2$ under constraint $y_i(w.x_i + b) \geq 1$. For a more relaxed constraints, we obtain the following optimization problem:

$$\text{Minimize } \Phi(w) = \frac{1}{2}\|w\|^2 + C\sum_i \varepsilon_i$$

$$\text{Subject to : } y_i(w.x_i + b) \geq 1 - \varepsilon_i, \varepsilon_i > 0, \forall i.$$

Large regularization parameter C involves high penalty for constraint violation. A possible solution for this problem, is to re-write C in terms of positive lagrangian multipliers α_i [164]. So the dual representation of the previous equation requires maximizing the following optimization problem:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i . x_j)$$

$$\text{Subject to:} 0 \leq \alpha_i \leq C, \sum_i \alpha_i y_i = 0 \text{ and yields}$$

$$w = \sum_{i=1}^{N} \alpha_i x_i y_i$$

Results of this optimization problem is thus a set of α_i^* coefficients so that the previous equation is maximized. These coefficients will be used to construct the optimal hyperplane. This latter can be seen as a linear combination of training examples. Only examples with $\alpha_i \geq 0$ are useful, they correspond to support vectors having a minimal distance to the hyperplane; the bias b is calculated by taking two arbitrary examples from the (+) and the (-) classes.

We can say that the SVM technique is a supervised learning where a training and a testing set are used. The training phase amounts to the generation of support vectors which constitute the representative of the class (the most near points to the separation border between classes).

Classification using the SVM method, is conducted by resolving a problem of the dimension N where N is the size of the training set. To resolve such problem, operations on large dimension matrices should be conducted which make this method considerably slow. This is the main drawback of this classification method.

However, the optimization problem presented in the SVM method, necessarily reaches a global minimum and avoid falling into a local minimum as in the case of Neural Networks [114]. In the other hand, SVM techniques are binary, so for a multi-class problem, we should reduces this latter to multiple binary classification problems. Discrete data are also problematic, a supplementary step which consists to rename these variables is necessarily in order to have good classification results.

2.7 Instance-based learning

Instance-based learning is a kind of lazy methods which are based on maximum likelihood estimation. It is a family of learning algorithms that, instead of performing explicit generalization, compare an instance to classify to instances observed in the

training set, which have been stored in memory. The principle of instance-based learning is to transfer past experiences (cases) on a problem to give a solution to a present situation. The main advantage of instance-based learning, if compared to other classification techniques, is its ability for incremental learning. Because no explicit model is needed to be generated in the training phase, the update of a new instance have no effect on the relevance of the model. This instance can simply be added to past experiences. Other methods, such as decision trees, generally require the entire set of training data to be re-examined in this case.

The most well known algorithm in instance based learning is the k-Nearest Neighbour algorithm [44]. The work in [1] and [126] presented an overview of most used instance based learning classifiers. The K-NN algorithm is based on the idea that instances, in the training set, usually exist in the proximity of other instances having similar proprieties [42]. To classify a new instance, the k-NN assigns the instance to the class labeling its nearest neighbor instance observed in the training set.

The k-NN identifies the k nearest neighbor instances to the instance to classify and then votes for the class with the highest frequency. In the following, we give the general principle of an instance based learning algorithm:

Algorithm 2 Instance-based classifier (Training instances T_r, Test instance a_{ts})

$class(a_{ts}) = \emptyset$
find the k most nearest instances in the training set according to a distance metric
Class(a_{ts}) = most frequent class label of the k nearest instances
return Class(a_{ts})

A good distance measure is a measure that minimize the distance between similar instances and maximize the distance between instances with different classes. Given two instances a and b, many metrics have been proposed to measure to what extend a is close to b.

- $D(a,b) = (\sum_{i=1}^{M} |a_i - b_i|^r)^{1/p}$: denotes the Minkowsky distance.
- $D_M(a,b) = \sum_{i=1}^{M} |a_i - b_i|$: denotes the Manhattan distance.
- $D_E(a,b) = (\sum_{i=1}^{M} |a_i - b_i|^2)^{1/2}$: denotes the Euclidean distance.
- $D_{Cheb}(a,b) = \max_{i=1}^{M} |a_i - b_i|$: denotes the Chebychev distance.
- $D_{Camb}(a,b) = \sum_{i=1}^{M} \frac{|a_i - b_i|}{|a_i + b_i|}$: denotes the Camberra distance.

To improve the accuracy of instance-based learning, a variant of algorithms use weighted vote to define the importance of each neighbor on the decision to take. A survey of weighting algorithms can be found in [167].

The k-NN has showed its efficiency in many real world problems, however there are some limits according to this learning method:

i) It requires a large set of data.

ii) It is sensitive to the choice of the distance measure when classifying new instances.

iii) The large storage requirement in the classification step caused by the necessity to scan all training instances in order to compute the distance from a testing instance.

iv) The difficulty to choose the optimal parameter k.

In particular, the choice of the parameter k affects the efficiency of the k-NN algorithm for more than one raison: if instances, not similar to an example to classify, are in its proximity, these instances will win the vote which causes a false classification. Usually a large k is recommended. In the other hand, if the region defining a class of instances is so small so that instances from this class surrounding the example win the vote, a small k should be used.

The work in [167] studied the effect of the choice of parameter k in presence on noisy instances. In [136] the authors presented a study of the k-NN efficiency as a function of domain characteristics which includes the size of the training set, the number of relevant or irrelevant attributes, the degree of noise in the data and parameter k.

The major advantage of instance-based learning is the simplicity of the training phase. In deed, this method do not require to generate any model for this step, if compared to decision trees for example. In addition, instance-based learning is known by its rigidity and stability to training set updating. While a small change in this set could involve re-examining all training instances in order to learn a new decision tree, this change have'nt an important effect on instance-based learning. This characteristic enables it to be a incremental classification technique.

As we have already mentioned, the main drawback of instance-based learning is the high computational time in the classification step. Many heuristics have been proposed to deal with this problem. The idea is to filter irrelevant instances for classification [115]. We cite for example, the ICF algorithm proposed by [28], in which the authors showed that the number of relevant instances can be reduced until 80% without considerably affecting the classification accuracy.

2.8 Statistical learning methods

Contrary to ANNs, statistical approaches are characterized by generating an explicit probabilistic model in the training step. The produced model is then used to provide the probability, for a test instance, to belong to each class, rather than simply choosing a class in an obscure way. Linear Discriminating Analysis: LDA and Fisher linear discriminants are simple methods used in statistics and machine learning to find linear combination of attributes which best separates binary or multiple object classes [87].

Other techniques are based on the maximum entropy to estimate probability distributions on data. The principle of this technique is that if nothing is known, the distribution should be the most uniform that possible, i.e., having a maximum entropy. Bayesian Networks are the most known statistical learning algorithms.

2.8.1 Bayesian Networks

Bayesian Networks(BN), initially proposed by [139], are graphic probabilistic models which enables acquiring and modeling knowledge from data. Particularly adapted to deal with uncertainty, these models can be manually described by domain experts or also automatically generated by learning.

A Bayesian Network have the same architecture as a Possibilistic Network previously introduced in Chapter 1. It is a simple graph in which nodes represent domain variables, and arcs (the graph is thus directed) connecting these variables are attached to *conditional probabilities*. Bayesian Networks are represented by acyclic graphs without loop (DAG). The arcs represent relations between variables which are either deterministic, or probabilistic. Thus, observing one or more causes do not systematically imply effects which depend on it, but only modifies the probability of observing them. The main interest of the BNs is to take into account simultaneously the a priori knowledge of experts (in the graph) and experience contained in the data. More formally, BNs are concise representations of the joint probability distribution (JPD)$P(V)$ on a set of random variables $V = \{V_1, ..., V_n\}$. The authors in [139] define a BNs as a triplet $\langle$V, G, $P(V_i|U(V_i))$ $\rangle$ defined by:

- A set of random variables $V = \{V_1, ..., V_n\}$.
- An oriented acyclic graph (DAG) G, where each node is associated to a variable V.
- Conditional probability distribution $P(V_i|U(V_i))$ of each variable V given its immediate parent U in the graph G.

A BN propagates probability distributions through the graph. This propagation results from the application of probability chaining rule to joint probability distribution. More precisely, after propagation, the following distributions are obtained:

$P(V_1 = v_{i_1}^{(1)}, ..., V_n = v_{i_n}^{(n)})$

$= P(V_n = v_{i_n}^{(n)} | V_{n-1} = v_{i_{n-1}}^{(n-1)}, ..., V_1 = v_{i_1}^{(1)})$

$*P(V_{n-1} = v_{i_{n-1}}^{(n-1)} | V_{n-2} = v_{i_{n-2}}^{(n-2)}, ..., V_1 = v_{i_1}^{(1)})$

...

$*P(V_2 = v_{i_2}^{(2)} | V_1 = v_{i_1}^{(1)})$

$*P(V_1 = v_{i_1}^{(1)})$

The independence assumption, usually adopted, announces that each variable V_i is independent of each other variable. This assumption considerably simplifies the previous equation because, in this case, each variable V_i is only dependent on its immediate parent. This simplification enables to eliminate variables V_i in the conditional structure. The joint probability distribution is calculated as follows:

$$P(V_1, ..., V_n) = \prod_{i=1}^{n} P(V_i | U(V_i)) \tag{2.2}$$

In BNs, a child variable V_i is connected by an oriented arc to his immediate parent ($U(V_i)$). Arcs are drown from parent to child.

To model a Bayesian Network, usually a database should be given which is used to generate the graph characterizing conditional dependencies between variables. This is held in two phases:

- **Network training:** The main question, in this step, is to derive the structure of the DAG. Then it is necessary to define its parameters; probabilities associated to the network from the given data.
- **Bayesian inference:** From results given in the first step, the network propagates information inside its structure, allowing any interrogation and can provide, for each partial or complete state of the database, the occurrence probabilities of all possible values of each variable.

The main contribution of BNs, if compared to decision trees or neural networks, is certainly the capacity to take into account priori knowledge for the given problem. This knowledge is represented through the structural relation between random variables. The a priori expertise or knowledge of the domain can take various forms according to the node which represents it. Indeed a node can be the root without any parent, a leaf without any child, a cause or a direct effect of another node. BNs offer also a complete or a partial order on nodes which requires that a given node must obligatorily appear before another node in the network.

Construction of the BN structure by training

The structure of Bayesian Networks is defined by the set of arcs of the directed graph in the network. In some cases, the BN structure is provided by an expert. If it is not the case, this structure can be learned from complete or incomplete data. The search of the network structure is a difficult problem mainly because the search space is of an exponential size according to the number of variables. So the main question is how to derive the best structure of the BN?

There are two general approaches of BN structure construction by training [84]. The first one is based on searching and scoring methods, the second one is rather based on dependency analysis methods.

The first approach have a heuristic nature, it seeks for the best structure which adapts to the data. It starts with a disconnected graph, uses search methods to add arcs and tests by the use of a score if the new structure is better than the old one. In the second approach, the problem is seen differently. The algorithms of this approach try to discover the dependency of the data and then employ these dependencies to imply the structure. Each of these approaches have advantages and disadvantages. Generally the approach based on the dependency analysis is more effective for a network in which the structure is not too complex, but the majority of these algorithms require an exponential number of tests on conditional independence.

François and Leray [84] have developed a comparative study of algorithms for BN structure construction by training [130]. This study is related to algorithms MWST (maximum covering tree), PC, K2 and GS (gloutonne search). The authors affirm that, MWST algorithm gives a graph close to the origin graph, in spite of the fact that this method traverses only the poorer space of the tree. The heuristic PC also gives good performances. This method builds structures with few arcs, all of them are almost relevant. The K2 method is very fast and is often used in the literature. It remains however too sensitive to initialization. Two different scheduling give two different BNs. For a fixed order, K2 always finds the same graph. On the other hand by changing scheduling, the final graph changes radically. K2 is employed with the algorithm MWST in order to give good performances. The algorithm GS is also

robust with respect to the variation of the database size especially if it is initialized with the tree obtained by MWST.

Inference in Bayesian Networks

Bayesian Networks enables representing a set of random variables for which we know that a dependency relation could exist. Let V be this set of variables and P(V) the probability distribution on V. If a new information on one or more variables is given, then one would like to update knowledge represented in the BN through P(V) in the light of this new information. This updating, conducted by using the Bayes rule, is called the inference process. More formally, the inference in a BN amounts to calculating $P(V|\epsilon)$, i.e. the calculus of the posteriori probability of the network knowing ϵ.

The first algorithms used exact inference (in opposition to approximate) for Bayesian Networks and are proposed by [138] and [109]: it is an architecture based on transferring message between nodes. In this technique, to each node is associated a processor which sends messages, in an asynchronous way, to its neighbors until stability is reached in a finished number of steps. This method was extended to other types of networks to give the algorithm JLO. This method is also called junction tree algorithm developed by [120] and [106].

Another method, developed in [139] and [105], is called the cat-set-conditioning: it consists in instancing a number n of variables so that the remaining graph forms a tree. We carry out a propagation by messages on this tree, then a new instantiation is selected. We reiterates this process until all possible instantiations were used.

Another algorithm is initially proposed by Zhang and Poole in [177]. It is primarily the variable elimination based algorithm of Dechter [45] which eliminates by marginalization (i.e. integration) variables one after the others. An order in which variables must be marginalized is required as input to the algorithm called the elimination order. The complexity of the variable elimination algorithm could be estimated by the number of numerical addition and multiplication operations which it carries out. Finding an optimal elimination order is considered as a difficult NP problem.

The inference in unspecified networks is NP-difficult [41], the complexity of the inference can lead to prohibitory computing times for complex networks. It is impossible to directly calculate the probability distribution of a node or to carry out a more complex inference, for this raison, a new type of inference named approximate inference is introduced. The approximation methods seek to consider the complete probability distribution represented by the network, by carrying out random pullings with simple distributions. The two main families of approximate inference algorithms

are the Monte Carlo algorithm [124] and the variational algorithm [108].

2.8.2 Naive Bayesian Classifiers

The Naive Bayesian Classifier (NBC) is a classification technique based on Bayes rule and Bayesian networks. This method is mainly suitable for data with high dimension. Despite its simplicity, NBC can often outperform more sophisticated classification methods [117]. NBC can be seen as a Bayesian network connecting attribute nodes to class nodes in which predictive attributes are assumed to be conditionally independent given the class attribute (Figure 2.3). This hypothesis is usually called *independence hypothesis*. Even if this hypothesis cannot usually be justified for some data sets, it contributes to clearly simplify the classification task since it enables calculating conditional probabilities $p(a_i|c_j)$ for each attribute in a separate way.

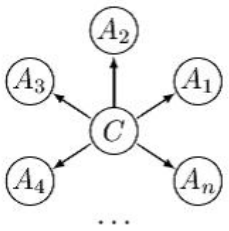

Figure 2.3: The NBC is a BN with a star structure

Even under the independence assumption, the NBC have shown good performance for datasets containing dependent attributes. Domingos and Pazzani [54] explain that attribute dependency does not strongly affect the classification accuracy. They also relate good performance of NBC to the zero-one loss function which considers that a classifier is successful when the maximum probability is assigned to the correct class (even if estimated probability is inaccurate). The work in [176] gives a deeper explanation about the reasons for which the efficiency of NBC is not affected by attribute dependency. The author shows that, even if attributes are strongly dependent (if we look at each pairs of attributes), the global dependencies among all attributes could be insignificant because dependencies may cancel each other out and so they do not affect classification.

Many Bayesian classifiers have been proposed, in the literature, to classify data with independent attributes in the case of discrete or continuous data [117][118][92] [88]. A semi-naive Bayesian classifier (SNBC) have been developed by [52] which is mainly suitable for discrete and continuous data. The SNBC takes into account correlation between attributes by grouping those which are very dependants and considering them as one attribute.

Existing approaches for Naive Bayesian Classifiers

A NBC can be defined by conditional probability distributions and the density function which estimates the dependency between attributes and classes. To estimate a density function for continuous data, principally three approaches are usually considered [141]:

- Discretize each continuous attribute and then estimate its probability distribution using a multinomial probability distribution.
- Directly estimate the density function in a parametric way by means of Gaussian densities.
- Directly estimate this density in a non parametric way using for example kernel density functions.
- Directly estimate this density in a semi parametric way using finite mixture models.

The most known Bayesian classification approach, in the literature, is that based on multinomial distribution estimation applied to discretized attributes. Since this first family of approaches supports only discrete variables, all continuous attributes must be discretized before classification. Certainly the discretized process contributes to an information loss. Even with discretization and the use of multinomial distributions, this family of classifier shows a high ability to estimate the true density when classifying new examples [33]. However, these approaches could be limited when we are faced to a classification problem having variables with many intervals because, in this case, the number of parameter to be estimated is relatively high. In such situation, the risk of overfitting should also be high. In the other hand, the number of examples used to estimate the distribution for each parameter become very low when attributes have many attribute values which involve to have a erroneous statistical calculus. This first family of Bayesian Classifiers is usually called *Multinomial Classifiers* (MC). A variety of multinomial classifiers are proposed to handle an arbitrary number of independent attributes. We cite for example, Naive Bayesian Classifiers [117][118][92], the tree-augmented naive Bayes classifier [88], the k-dependence Bayesian classifier (kDB)[152], the semi-naive Bayesian classifier: SNBC [112][52] and finally the Bayesian network-augmented naive Bayes [33].

The second family of approaches uses parametric densities to estimate the true density for variables. The most used densities are Gaussian functions which are, generally, considered as a good approximation for the true densities in many domains [107]. This approach is based on the normality assumption of attributes in the context of parents (classes). Such classifiers are named *Gaussian Classifiers* (GC)

since they combine Bayesian Networks with the normality assumption [91][140]. We note that these classifiers have fewer complexity difficulties, if compared to Multinomial classifiers, to model complex graphs. Moreover, parameter estimation is more robust because distributions are learned from an example set more large than that used in MCs. In fact, the training set is partitioned only according to class values i.e. in average, we have N/C examples for each estimation with N is the size of the training set and C is the number of class values.

When the true density is not too far from the Gaussian, GCs show a classification at least as good as MCs. Although the normality assumption may be a real approximation for many benchmarks, it is not always the best estimation. A new family of approaches, using non parametric approximation, have been proposed to overcome limits of GCs and to improve their efficiency.

The non parametric approaches abandon the normality assumption and use instead a kernel density based estimation. The derived classifier is named *Flexible Classifier*. This nomenclature is due to the ability of such classifier to represent densities with more than one mode if compared to a simple Gaussian classifier. Flexible classifiers represent densities of different shapes with high accuracy; however they contribute to a considerable increase in complexity. In fact, this family of approaches is able to model real densities with many shapes with a high precision. Kernel densities are more flexible than Gaussians when modeling densities with many modes.

The semi parametric approaches can be seen as a second alternative to break with the strong parametric assumption. These methods combine the advantages of both parametric and non parametric family approaches. The semi parametric methods are not limited to a particular shape of the density function. The training process, in semi parametric models, is more complex than the simple process used in parametric or non parametric.

The most used non parametric methods are the mixture models [80][129], the Gaussian mixture models [14] [129] and the kernel density based approaches [107]. Pérez et al. [141] have recently proposed a new approach for Flexible Bayesian Classifiers based on kernel density estimation that extends the FNBC proposed by [107] in order to handle dependent attributes and abandon the independence assumption. In this work, three classifiers: tree-augmented naive Bays, a k-dependence Bayesian classifier and a complete graph are adapted to support kernel Bayesian network paradigm.

The main drawbacks of Bayesian classifiers is that they are not appropriate for databases with many attributes. In this case the network construction become rapidly difficult in terms of time and space.

Principle of Naive Bayesian Classifiers (NBC)

Given a new vector $a = \{a_1, a_2, ..., a_M\}$ to classify, a NBC calculates the posterior probability for each possible class $c_j (j = 1, ..., C)$ and labels the vector a with the class c_j that achieves the highest posterior probability, that is:

$$c^* = arg \max_{c_j} p(c_j|a) \tag{2.3}$$

Using the Bayes rule:

$$P(c_j|a_1, a_2, ..., a_M) = \frac{P(a_1, a_2, ..., a_M|c_j) * P(c_j)}{P(a_1, a_2, ..., a_M)} \tag{2.4}$$

The denominator $P(a_1, a_2, ..., a_M)$ is a normalizing factor that can be ignored when determining the maximum posterior probability of a class, as it does not depend on the class. The key term in equation 2.4 is $P(a_1, a_2, ..., a_M|c_j)$ which is estimated from training data. Since Naive Bayes assumes that conditional probabilities of attributes are statistically independent we can decompose the likelihood to a product of terms:

$$P(a_1, a_2, ..., a_M|c_j) = \prod_{i=1}^{M} p(a_i|c_j) \tag{2.5}$$

In the continuous case, each numerical attribute is modeled in different ways using normal, uniform or gamma density functions.

A supplementary common assumption made by the NBC in this case, is that within each class the values of numeric attributes are normally distributed around the mean and so models each attribute through a single Gaussian. The NBC represent such distribution in terms of its *mean* and *standard deviation* and compute the probability of an observed value from such estimates. This probability is calculated as follow:

$$p(a_i|c_j) = g(a_i, \mu_j, \sigma_j) = \frac{1}{\sqrt{2\Pi}\sigma_j} e^{-\frac{(a_i-\mu_j)^2}{2\sigma_j^2}} \tag{2.6}$$

The Gaussian classifiers [91][107] are known by their simplicity and have a fewer complexity if compared to other non-parametric approximation. Although the normality assumption may be a real estimation for many real world problems, it is not always the best estimation. Moreover, if the normality assumption is violated, classification results of NBC may deteriorate.

John and Langley [107] proposed a Flexible Naive Bayesian Classifier (FNBC) that abandons the normality assumption and instead use nonparametric kernel density estimation for each conditional distribution. The FNBC has the same properties as those introduced for the NBC, the only difference is instead of estimating the density for each continuous attribute a_i by a single Gaussian $g(a_i, \mu_j, \sigma_j)$, this density is estimated using a averaged large set of Gaussian kernels. To compute continuous attribute density for a specific class j, FNBC calculates n Gaussians each of them stores each attribute value encountered during training for this class and then takes the average of the n Gaussians to estimate $p(a_i|c_j)$. More formally, probability distribution is estimated as follow:

$$p(a_i|c_j) = \frac{1}{N_j} \sum_{k=1}^{N_j} g(a_i, \mu_{ik}, \sigma_j) \tag{2.7}$$

where k ranges over the training set of attribute a_i in class c_j, N_j is the number of instances belonging to the class c_j. The mean μ_{ik} is equal to the real value of attribute i of the instance k belonging to the class j, e.g. $\mu_{ik} = a_{ik}$. For each class j, FNBC estimates this standard deviation by:

$$\sigma_j = \frac{1}{\sqrt{N_j}} \tag{2.8}$$

The authors also prove kernel estimation consistency using (2.8), (see [107] for details). It has been shown that the kernel density estimation used in the FNBC and applied on several datasets, enables this classifier to perform well in datasets where the parametric assumption is violated with little cost for datasets where it holds.

2.9 Evaluation methods of classifiers

It is important to evaluate classifier performance in order to determine whether to employ this classifier for a given situation. For example when learning the effectiveness of medical treatments from a limited-size data, it is important to estimate the error rate of the classifier.

In general, evaluation methods can differ by which metric(s) they use for measuring performance. The most common metrics are *Accuracy* (the number of correct classification examples) or *Error rate* (the number of incorrect classifications). These metrics are either used alone or combined with other metrics, e.g., some methods combine accuracy with classifier *complexity* metric. Using complexity to evaluate a classifier performance is justified by the fact that if we have two equally good solutions we should pick the simplest one.

Some other methods for using the data to estimate and compare classifier accuracy may also be used. To compare two classifiers to each others, more advanced statistical tests can be used such us: Student t-test, Signed-Ranks test, McNemar,... The common way used for these tests is to check whether the difference between two classifiers results over various data sets is non-random (significantly different from zero). For this purpose, a statistical hypothesis test h should be formulated:

- Claim that both classifiers have equally good performance.

- If claim is rejected, we accept that the first classifier is better.

To accept or reject such hypothesis, we need to compute some statistical metrics which are specific to each typical test. A complete review of statistical comparison between classifiers is given by Demsar [50].

2.9.1 Accuracy

The simplest way to evaluate a classifier is *accuracy*: it is the proportion of instances whose class the classifier can correctly predict. To do this we need a dataset that contains instances whose class already known. We ask the classifier to predict the class of each of these instances in turn. Then we compare the classifier prediction with the real class of each instance. We estimate the accuracy by the *Percent of Correct Classification*(PCC) defined as follows:

$$PCC = \frac{number\ of\ well\ classified\ instances}{total\ number\ of\ classified\ instances} * 100 \tag{2.9}$$

Calculating accuracy using the same dataset that we used to build the classifier is not a proper way to evaluate the classifier. Its accuracy on examples it has already seen (the ones stored in the memory of a kNN classifier for example) could be an optimistic estimate of its future performance. Thus in classification problems, we try to find a good generalization of the classifier, i.e. to what extend it is able to perform on unseen examples that will be presented to it in future when it is applied to real data.

In order to guarantee this generalization ability, the classifier should be build using one dataset, what we have called training set and evaluated it on different dataset, usually called testing set which is a set of independent instances not used for building the classifier. For these two datasets, we need : i) instances to be already classified, ii) the sample to be representative of the real population, and iii) ideally large number of examples in the training set in order to have a good classifier an in the testing set in order to have more reliable estimate of accuracy.

2.9.2 Testing strategies

In real world problems, it is always difficult to collect two large, independent and representative datasets of already-classified instances. A common way is to collect one large dataset and then seek for many ways to construct the training and testing sets from this original dataset. We give here some of these methods.

The holdout validation

The simplest method is to partition the original dataset into two randomly selecting instances for a training set (usually 2/3 of the original dataset) and a test set (1/3 of the dataset). The classifier is build using the training set and then evaluated on the remaining testing set. This method has the advantage of being simple. But the problem is that we cannot be sure about the representativeness of each dataset and consequently we can just be in a very optimistic or pessimistic case.

This method can be considerably improved by repeating it several times. In each time, the training and testing set are randomly selected. The overall accuracy is estimated by averaging accuracies obtained in each time.

Cross Validation

In k-fold cross-validation, the data is divided into k subsets of equal size. The classifier is trained k times, in each time only one of the k subsets is used for testing (computing accuracy) and the remaining $k-1$ subsets for training. The overall accuracy is obtained by averaging the k elementary accuracies. A value of 10 for k is commonly used. Cross validation is a good estimate of accuracy especially for datasets with small size. It allows the use of all of the data for training.

Leave-one-out cross-validation

This method is a particular case of k-fold cross-validation in which $k = n$, where n is the size of the original dataset. Thus from this dataset only a single instance is used for test and all remaining $n-1$ instances for training. Repeating this process for all instances will guarantee that each of these instances is used for testing and avoid the problems of random selections but this method has a higher complexity if compared to the k-fold cross-validation.

2.10 Comparative study of classification methods

Machine learning techniques can be useful in several domains. Each technique have its particular advantages and drawbacks. An important question for a decision maker is "which algorithm is the most appropriate for a given situation ?". In order to have a convenient response to this question, we try to discuss in this section positive and negative points for each classification technique and their relation to real world problems.

To conduct this comparative study, we are based on the study done in [114]. The Table 2.1 includes author appreciation against these techniques according to the most known criterions in the literature. The evaluation consists to assign a particular number of stars for each technique. In this comparison, (****) stars represent the best and (*) star the worst performance.

Table 2.1: Indexation of criterions

Criteria Indexes	**Description of the criteria**
CA	Classification accuracy
SL	Speed of learning
SC	Speed of classification
TMA	Tolerance to missing attributes
TIA	Tolerance to irrelevant attributes
TRA	Tolerance to redundant attributes
TIA	Tolerance to highly independent attributes
TD	Type of data (discrete, binary, continuous)
TN	Tolerance to noise
OR	Dealing with overfitting risk
IL	Incremental learning ability
EA	Explanation ability / transparency of knowledge
MP	Model parameter handling

By analyzing evaluation results in Table 2.1 and 2.2, we can see that the NN and SVM are similar in more than one point as DT and rule learner are. We also note that NBC and K-NN have their particular characteristics. Overall, Neural Networks and SVM are the most efficient methods in terms of classification accuracy whereas Bayesian Networks seems to be the least accurate. However this evaluation corresponds only to classical NBCs and do not consider new extensions of this technique such as Flexible Naive Bayesian Classifiers (FNBC) which are largely better than the classical NBC, as will be shown in Chapter 5.

The NN and SVM are more appropriate for classification of continuous attributes and data with high dimensionality (large number of attributes and instances). For these two techniques, a large set of training examples should be given in order to reach the highest prediction accuracy. We can also note that these methods are

Table 2.2: Comparative evaluation of classification algorithms

	DTs	**NN**	**NBCs**	**kNN**	**SVM**	**RL**
CA	**	***	*	**	****	**
SL	***	*	****	****	*	**
SC	****	****	****	*	****	****
TMA	***	*	****	*	**	**
TIA	***	*	**	**	****	**
TRA	**	**	*	**	***	**
TIA	**	***	*	*	***	**
TD	****	***	****	***	**	***
TN	**	**	***	*	**	*
OR	**	*	***	***	**	**
IL	**	***	****	****	**	*
EA	****	*	****	**	*	****
MP	***	*	****	***	*	***

principally efficient to deal with problems having a non linear relation between inputs and outputs.

As we have already mentioned, the SVM has the limit of being a binary algorithm, a pre-treatment step is necessary in order to reduce a multi-class problem to more than one two-class problems. NN and SVM have mainly two important drawbacks, not only they are too slow in the training phase but also are seen as "Black boxes" because of their weak interpretation capacity. Several approaches are interested to extract IF-THEN rules from NN in order to give for this technique more comprehensibility. A complete study for rule extraction from NN can be found in [76].

In contrary, DTs and Rule-base learning are models with high interpretation ability and are very quick in training if compared to NN and SVM. This clearly show a high complementarity between these two families of methods (DTs and NN for example) which explains hybridizing between these methods in a variety of neuro-symbolic systems, we cite for example [94] [95].

It is also known that inductive methods are well suitable for discrete data. A discretization phase of continuous data is necessary which leads, in general, to loss of information. In addition, algorithms for learning DTs divide example space orthogonally to the axe specific to one attribute and parallels to all other axes. This process prevent DTs to be appropriate for problems where a diagonal partitioning of data is recommended. Moreover, the pruning procedure described in DTs and also rule-based learners enables to these techniques to be more resistant to data errors if compared to kNN technique for example.

On the other hand, algorithms for instance-based learning are known as lazy

methods. Despite the negligible training step, these methods makes a large time in classification. To classify each test example, the kNN algorithm scans all training examples to calculate distances. In contrary, NBC seems to be the most rapid, among all other techniques, as well as in training and classification. In fact this technique requires only one scan on the data to calculate probability distributions as well as in the discrete or continuous case.

We usually reproach to instance-based learning of being very sensitive to noisy and irrelevant data. The similarity measure become simply erroneous in presence of a noise in one attribute value which leads to falsify the process of searching for the nearest neighbor and so incorrectly classifies the test example. This technique has a weak interpretation capacity principally when the nearest instances, used for classification, are very heterogeneous. In such situation, we cannot be able to extract any interpretation from them mainly if these instances are numerous (large k). However, instance-based learning is transparent and can be simply implemented because it is totally inspired from human reasoning.

The Naive Bayes technique has the advantage of requiring a limited memory space in the training and testing phases: The only necessary space is that for saving priori and conditional probabilities. In the contrary, kNN requires a more large memory space in classification which should be as large as the training set size. It should be noted that NBC and kNN have one common characteristic: they are viewed as powerful techniques to deal with incremental training (when the training set is not very representative of the real world data) whereas induction algorithms (such as DTs) are too limited in this context.

To all previously cited advantages of NBCs, we add that these classifiers are very robust to deal with incomplete attributes. While these attributes can be simply ignored when estimating conditional distributions in NBC, the kNN requires all attribute values to be completely specified to do classification. This characteristic enables to NBCs to be the *best* classifier to use in presence of data uncertainty or imperfection.

However, NBCs presents some pitfalls. In particular, they assume that a data set could simply be represented through a unique probability distribution and that this statistical model will be sufficiently able to generalize and to discriminate between classes. This justifies, in the first hand, low classification accuracy of classical NBC, if compared to other technique generating more complex models such as DTs, and largely better classification results of the FNBC (based on kernel distributions) in the other hand. NBCs are also viewed as transparent models very easy to exploit in many domains such as physics and mechanics to reflect human diagnostics.

This comparative study shows that the NBC may be a promising technique which is advantageous in more than one sight and could be improved (mainly the limiting

criterions) if more attention is given in order to estimate the convenient probability distribution.

In this chapter and in the frame of this thesis, we are principally interested to Bayesian Classifiers in their classical and extended versions. Various advantages of NBCs give as the intuition to more develop these models which seem to be promising if faced to imperfect data which is, actually, a serious problem in many real world domains.

2.11 Conclusion

In this chapter, we briefly presented the most used classification techniques and we focused more on Bayesian classifiers. Advantages and limits of each method are also studied. In this dissertation, we are mainly interested to statistical methods based on Bayesian networks. Our first focus is to study the possibility of extending these classifiers, known by their simplicity and efficiency, to support uncertainty in data representation.

A major classification problems, usually encountered in real world databases, is how to deal with imperfect data in the training/testing sets?. Most of existing methods fails to accomplish classification when they are faced with uncertain or imprecise data. A common way to deal with such case is to clean imperfect instances before conducting classification. This heuristic reinforces generating incoherent models in context of the available data. In contrary, there is a high necessity to adapt classification techniques to be more able to treat and support uncertain data instead of going through rejection heuristic because uncertain data could contain important knowledge about the domain. In the third chapter we present a complete discuss of rule induction methods since the first contribution in this work focuses on this classification technique.

Chapter 3

Rule-based learning

3.1 Introduction

Rule induction holds a privileged place in machine learning techniques as it provides many benefits to decision makers. In fact, it ensures an easily interpretable predictive model. Furthermore, this method produces such a model without any preliminary statistical knowledge. The produced model include a set of rules of the form "IF condition THEN Conclusion" (e.g. If the bank customer is faithful and its salary is high Then the decision is to grant the credit"). On the other hand, generated rules can easily be implemented in information systems (e.g. translation of rules to SQL queries).

In this chapter, we focus on supervised rule-based learning. Among rule induction methods, we are interested to the "separate-and-conquer" based approach, which monopolized the methods of machine learning in the Nineties.

The second family of approaches for rule induction is the decision tree technique previously introduced in Chapter 2. Indeed, a decision tree can be translated to a set of rules where each branch of the tree, starting from the root node and going to the leaves, constitutes a rule having as premise all conditions (tests) in this branch.

To this list of methods is added approaches based on rule extraction from neural networks. These latter have been proposed to answer criticisms made to connexionist methods seen as "black boxes". Techniques for rule extraction from neural networks have the capacity to improve the legibility of artificial neural networks without decreasing its high classification accuracy. This family of methods will not be detailed in this context. An excellent review of most existing approaches for rule extraction from neural network is given in Duch et al. [76].

In the following section, we first give a general presentation for rule based classifiers without any technical details. Section 3 recapitulates the decision trees for rule generation, where we focused on ID3 and C4.5 algorithms as basic algorithms to produce a decision tree model. In this section we also present the method for rule extraction from a decision tree. In Section 2, we present the background of sequential covering method used in the literature. In this context, we first introduce the general sequential algorithm proposed by Fürnkranz (1999)[89] and then present a variety of rule induction algorithms based on this method. The PART algorithm is presented in Section 4, an illustrative example is also given.

Through the different sections of this chapter, we give a discuss of the presented rule-based learning approaches. This analysis enables us to propose, in Chapter 4, a new approach for evaluation and selection of classification rules seen as an attempt to solve some problems countered with rule-based learning.

3.2 Rule representation

A rule-based classifier represents a function mapping examples, described by a set of attributes (features), to target classes (output concept). Rules considered in such problems are called classification rules and they are expressed in the form: IF (antecedent) THEN (class), where the antecedent part is formed by a conjunction of elementary tests on values of attributes and the class part indicates the assignment of a example, which satisfies the antecedent part, to the given class label.

Although the form of a decision rule may differ if we deal with crisp or fuzzy paradigm, they always partition the whole feature space into some sub-spaces. A decision rule r assigning examples to a class c is represented in the form:

$$r : If(A_1\ op\ a_1) \wedge (A_2\ op\ a_2) \wedge ... \wedge (A_M\ op\ a_M) \text{ then } c$$

Where $(A_1\ op\ a_1) \wedge (A_2\ op\ a_2) \wedge ... \wedge (A_M\ op\ a_M)$ is the antecedent part of r and c is its class label. The antecedent part is a conjunction of elementary conditions $(A_i\ op\ a_i)$. Each condition represents a test on a value of a corresponding attribute. For a symbolic attribute, the test compares its value to a constant (op is the equality), and for numerical attributes other comparisons relations are rather used (e.g., $op \in \{=, \leq, \geq, <, >\}$)

Rules induced for classification are represented in a disjunctive normal form, $R = (r_1 \vee r_2 \vee ... \vee r_k)$, where R is known as the rule set forming the induced classifier and r_i's are the classification rules or disjuncts presented by a set of conjunctions.

A training set T_r is usually used for learning the set of rules R. For a given class c , training examples from this class are called its positive examples, while examples belonging to the remaining classes are called negative examples of c.

In the following, we briefly present some definitions of basic decision rule properties:

- A rule r **covers** an example if it matches the condition part of this rule. We also say that the rule is satisfied or fulfilled by the given example. The set of all examples, which descriptions (attribute values) satisfy elementary conjuncts of the rule, is called the *covering set* of r.
- The rule set R **completely covers** all positive examples of class c, if each of these examples is covered by at least one decision rule from R.
- Furthermore, we say that R is the **minimal cover** of c if there is no other $R' \subset R$ that covers all positive examples of c. This means that R completely represents positive examples of this class by the smallest number of rules.

Given the training set T_r and a decision rule r, two quality measures are usually used to evaluate a decision rule:

- **The coverage** of the rule is defined by the number of examples in T_r that satisfies the antecedent part of the rule divided by the number of examples in T_r.
- **The accuracy** also called $confidence\ factor$ is defined by the number of examples coved by r whose class labels are equal to c, i.e. the examples satisfying rule antecedent and consequent at the same time. More formally these measures are defined by:

$$Coverage(r) = \frac{\mid T_r^{cover} \mid}{\mid T_r \mid} \tag{3.1}$$

$$Rule - accuracy(r) = \frac{\mid T_r^{cover \cap correct} \mid}{\mid T_r^{cover} \mid} \tag{3.2}$$

where $\mid T_r^{cover} \mid$ is the cardinal of examples covered by the rule r and $\mid T_r^{cover \cap correct} \mid$ is the cardinal of examples covered by r and having the same class label c.

3.2.1 How to classify a new example using a rule-based classifier ?

A rule-based classifier classifies a test example using the rule covering this example. To illustrate how a rule-based classifier works, let consider the following example:

Example 3.1.

The iris data set [131] contains 150 examples each one is a flower (iris) of one of the different categories: $'Iris - setosa'$, $'Iris - versicolor'$ and $'Iris - virginica'$. Each category represents the class of the example. For each example, 4 numerical attributes are assessed: the $sepallength(SL)$, $sepalwidth(SW)$, $petallength(PL)$ and $petalwidth(PW)$ of iris plants. In this classification problem, the goal is to determine what distinguishes each category of iris plants from one another so that it is possible to know to which category an iris plant belongs given the four input variables. Assume that the following set of rules is generated using a rule based learning method.

$$r_1 : If(petalwidth \leq 0.6) \rightarrow' Iris - setosa'$$

$$r_2 : If(petalwidth \leq 1.7) \wedge (petallength \leq 4.9) \rightarrow' Iris - versicolor'$$

Table 3.1: Testing set

Instance	**SL**	**SW**	**PL**	**PW**	**Class**
1	6.9	3.1	4.2	1.5	**?**
2	5.1	3.5	1.8	0.2	**?**
3	5.5	2.3	6.1	1.9	**?**

Let consider the three test examples to classify given in Table 3.1. By inferring the two previous rules for each test example, we can see that:

- The first iris, having petal width equal to 1.5 and a petal length equal to 4.2 triggers only the rule r_2, and thus is classified as $'Iris - versicolor'$.
- The second iris, triggers both rules r_1 and r_2. Since the classes predicted by the two rules are contradictory ($'Iris - setosa'$ and $'Iris - versicolor'$), this rises a conflicting class problem usually denoted by the multiple classification problem.
- None of the two rules can be fired if confronted to the third iris. This case shows that the rule set may not cover some testing examples. The produced classifier should be able to classify test examples not covered by any rule.

To avoid the two last problems, the rule set should be **mutually exclusive** and **exhaustive**. Rules in a rule set R are mutually exclusive if no two rules in R are satisfied by the same example which means that no conflict between rules can be detected. The rule set R is exhaustive if all examples are covered by at least one rule in R. A rule set satisfying these two properties, can avoid the multiple classification and non-covering problems at the same time.

To deal with the non-covering case, most rule-based classifiers add a default class (a rule with an empty antecedent) to cover the remaining examples. This rule is triggered if all other rules are not satisfied. A common way to choose the default class, is to use the *largest class* covering the large set of examples or the *majority class* of examples not covered by existing rules. To overcome the multiple classification problem, two ways are usually applied: ordered versus unordered rule based classifiers.

3.2.2 Ordered rule-based classifiers

Rules are ordered in decreasing order of their priority, e.g. in a descending accuracy or coverage order. Then a test example is classified by the *first matching rule* in the rule set (that covers the example). So even if there are further rules covering this example, only the first one is triggered. This process will avoid the multiple classification problem. The rule set is usually tailed by a default class used to classify each non covered example. Ordered rule-based classifiers are also named **decision lists**.

Applying the ordered rule principle for the previous iris example, the second instance will be classified as 'Iris- setosa' whereas the third one can be assigned to a default class covering the 'Iris-virginica'.

3.2.3 Unordered rule-based classifiers

This approach supports the multiple classification case where a test example can be satisfied by many rules and considers the consequent of each satisfied rule as a **vote** for a particular class. The vote strategy enables to select a class label for the test example. The class with the highest number of votes is usually assigned to this example. Voting is usually based on weighted rules (using accuracy for instance). Unordered rules have some advantages if compared to decision lists. In particular, rules are considered in a symmetric way by giving the same chance for each rule to be triggered however in decision lists, some rules are prioritized which may reinforce neglecting some other relevant rules. That's way, unordered rules are assumed to be less sensitive to errors caused by the choice of the wrong rule used for classification

while ordered rules are too related to the rule ordering method. Furthermore, unordered rule-based classifiers have a little cost in the building phase because there is no need to sorting rules with a supplementary cost in the classification phase since a test example should be compared to each rule in the rule set.

3.3 Decision trees for rule generation

We first present the background for this method, before going up to rule generation from decision trees. As previously presented in Chapter 2, the most basic algorithm for decision tree construction is the ID3 and its successor C4.5. Thus we first give the principle for these two algorithms and then develop the method for extracting rules from DTs.

3.3.1 The ID3 Algorithm

ID3 builds a decision tree in a recursive way by choosing the best attribute for separation. This algorithm uses exclusively categorical attributes and a node is created for each value of the selected attribute. ID3 is a basic algorithm which is easy to implement, however the generated decision trees are neither robust, nor compact what makes them not adapted to large datasets. ID3 builds the tree by dividing the training set, at each node, according to a criterion depending only on one attribute.

The first step for building the decision tree consists at choosing the attribute that will be used for the first partitioning (the first division node or test). To build each node, one can use a strategy which best separates between classes. The initial junction produces several nodes that will be divided, in the same way as the root node, by searching a candidate attribute for a new partitioning. If an enumerative attribute were used for division in a higher level of the tree and since it can take only one value, thus it will be removed from the list of candidate attributes for a new division or branching. A node is considered as a leaf if we cannot find a supplementary partitioning that decreases the heterogeneity of this node.

At the end of the building process, a T_{max} tree is produced. Each leaf in this tree, covers only pure data which reflects an exact learning of the given training examples.

The basic ID3 algorithm

The main idea of this algorithm is to build a classification tree with a null error rate. When the tree is partially built, each node covers a subset of training examples:

those which satisfy all tests leading to this node. The building process should be continued if this subset contains examples from different classes. At each iteration, the best attribute is chosen for partitioning.

Algorithm 3 ID3 Algorithm

```
if all instances belong to only one class then
  The decision node is a leaf labeled by that class
else
  repeat
    for each generated subset T_r do
      Select the attribute that maximizes Gain(T_r, A_k)
      Split T_r into several subsets, one for each value of the selected attribute.
    end for
  until One of the stopping criteria is satisfied
end if
```

The choice of the best attribute (Splitting strategy)

ID3 starts by deciding which attribute will make the best partitioning. This latter is defined as that which best separates the current training set into groups in a way they are covered by the same class. The partitioning strategy used in ID3 is the *Information Gain*. This strategy is based on the information theory that can be explained as follows:

In general, we note that the necessary number of bytes required to describe a situation or a result depends on the size of the possible result set. For example, if there are eight possible classes of equal probabilities, we need $log_2(8)$ or 3 bits to identify one of them. On the other hand, if there are only four classes, we only need $log_2(4)$ or 2 bits. Thus, a division starting from a node whose examples belong to eight classes and producing new nodes whose examples belong to four classes in average, has an information gain equal to: $log_2(8) - log_2(4)$ or one bit.

The information size necessary for representing a random variable belonging to a certain class is: $-Log_2(P(C = c_j))$ bits, with C is a discrete random variable describing the class attribute and c_j is its value.

Based on these definitions, the average information quantity necessary to identify the class of a given example in a training set denoted *entropy* can be deduced as follows:

$$Info(T_r) = -\sum_{c_j \in C} P(C = c_j) log_2 P(C = c_j) bits \tag{3.3}$$

The quantity $P(C = c_j)$ is estimated by the frequency of the class c_j; i.e: $freq(c_j) = \frac{|T_r^j|}{|T_r|}$, where $\mid T_r^j \mid$ represents the number of examples in the training set T_r labeled with the class c_j.

The entropy can also be measured after splitting the set T_r under several subsets using the partitioning results carried according to attribute A_i:

$$Info(T_r|A_i) = -\sum_{a_i \in D_{A_i}} \frac{\mid T_r^{a_i} \mid}{\mid T_r \mid} Info(T_r^{a_i}) \tag{3.4}$$

Where $T_r^{a_i}$ denotes the training subset containing examples whose attribute value is a_i.

Using these two measurements, one can define the expected reduction in entropy (impurity) after partitioning according to A_i also known as the *information gain* defined by:

$$Gain(T_r, A_i) = Info(T_r) - Info(T_r|A_i) \tag{3.5}$$

At each node, the training subset of T_r is partitioned according to the attribute that maximized the gain.

3.3.2 The C4.5 Algorithm

This algorithm have been proposed by Quinlan 1993 [145] in order to palliate limits of the ID3 algorithm. Since the C4.5 is completely reposed on the ID3, we will restrict this section to give principal improvement provided by C4.5.

In particular, C4.5 uses an elaborate strategy: the "*Gain Ratio* " in order to limit the tree partitioning by penalizing attributes leading to many branching (with many different values). The C4.5 also gives more flexibility for the use of input data sets. The main improvements, is the possibility of using:

- Continuous attribute values.
- Grouping a set of discrete or nominal attribute values in order to support more complex tests.
- The NULL value.
- Missing attribute values.

Possible Candidate Tests

C4.5 uses tests of three types, each one involves only a single attribute A_i. Decision regions in the instance space are thus bounded by hyperplanes, each of them is orthogonal to one of the attribute axes.

- If A_i is a discrete attribute with v different values, v possible tests of the form "$A_i = a_i$" can be defined one for each value a_i of this attribute.
- More complex tests for discrete values can also be used in the form: $A_i \in S$ with $2 \leq S \leq v$ outputs, where $S = \{S_1, S_2, ..., S_s\}$ is a partition set of the values of attribute A_i. To find such tests C4.5 uses a greedy search and selects the partition set S that maximizes the value of the splitting criteria (discussed below).
- If A_i has numeric values, the form of the test is "$A_i \leq \theta$" with a boolean result true or false, where θ is a constant threshold. To find possible values of θ, all distinct values a_i of attribute A_i that are observed in the training set T_r are sorted and then one threshold is identified between each pair of adjacent values. (So, if we have v distinct values for A_i in T_r, $v - 1$ thresholds can be considered.)

The partitioning strategy (Gain Ratio)

The C4.5 is mainly based on the divide and conquer algorithm, where the most focus is to choose the best test at each step. In this algorithm any test on an attribute A_i that partitions T_r non-trivially will produce a decision tree, but different tests give different trees.

A serious challenge of decision tree algorithms is to produce models as simple as possible because a complex tree does not help for a good understanding. Such complex decision trees may also have weak predictive accuracy [150]. As an attempt to optimize the size of the tree, C4.5 selects the best candidate attribute, at each step, using a greedy search. For this purpose, the *gain ratio* criterion is used as an attribute selection heuristic. This criterion is defined by:

$$SplitInfo(T_r, A_i) = -\sum_{a_i \in D(A_i)} \left(\frac{\mid T_r^{a_i} \mid}{\mid T_r \mid} * log_2\left(\frac{\mid T_r^{a_i} \mid}{\mid T_r \mid}\right)\right) \tag{3.6}$$

$SplitInfo(T_r, A_i)$ measures the information obtained after dividing the set T_r into v sub sets according to the different values of the attribute A_i.

$$GainRatio(T_r, A_i) = \frac{Gain(T_r, A_i)}{SplitInfo(T_r, A_i)} \tag{3.7}$$

After evaluating all possible attribute tests, C4.5 selects the one that maximizes the value of *gain ratio*.

In the particular case of continuous attribute values, the algorithm searches for the best threshold to separate independent numerical values observed in the training data. The idea is to pick a threshold that have the highest gain ratio. The algorithm starts by sorting numeric attribute values, that appear in the training set, and then identifying adjacent examples that differ in their class labels, this enables to produce a set of candidate thresholds. Finally the gain ratio is computed for each candidate and choose the best one for splitting.

The tree pruning

Once the initial decision tree is constructed, a pruning procedure should be performed in order to decrease the overall tree size and decrease the error rate made by the tree [145]. Tree pruning is an important part of decision tree construction as it is used to improve the prediction accuracy by ensuring that the constructed tree model do not overfit the data set. The strategy used by C4.5 for pruning is based on the *prediction error rate* criteria.

C4.5 eliminates the branches from the tree by using a pessimistic elimination method. Tree pruning involves replacing a whole sub-tree by a leaf. Substitution occurs if a given decision rule shows that the expecting error rate in this sub-tree is higher than those of a simple leaf.

3.3.3 Rule extraction from a Decision Tree

Decision tree models are known by their comprehensibility and efficiency in the classification task. The C4.5 is the most popular algorithm for decision tree induction since it is widely available and reasonably well tested.

However for some data sets, especially when categorical attributes have many values, decision tree model can become quickly large and so losses its interpretation ability. A common way to overcome this limit is to transform the decision tree into a rule set. In this section, we review the method, used by the C4.5, to build a rule-based classifier by extracting IF-THEN rules from the tree model. In compared to a decision tree, IF-THEN rules may be easier for humans to understand, especially if the decision tree is very complex.

Rule Generation

To extract rules from a decision tree, one rule is created for each path from the root to a leaf node. Each attribute test given by a node in each path, becomes a conjunct in the rule antecedent; all conjuncts are logically combined using the logical AND. The leaf node in each path forms the consequent part of the rule.

Generated rules are then grouped in a disjunctive way using the logical OR. All rules extracted in this way are exhaustive and mutually exclusive. They are exhaustive since all examples are covered by at least one rule in the rule set, so no default rule is needed to be defined and rules can be triggered in unordered manner. Rules extracted from the tree model are also mutually exclusive since any example can be satisfied by only one rule which means that no conflict between rules can be observed. This prevents the multiple classification problem.

Rule pruning

Most rules extracted from a decision tree, following this rule generation process, are also too complex and even more difficult to interpret. The set of rules can be significantly simplified without decreasing their classification ability. In fact, the decision tree model usually contains unnecessarily branching which leads to a set of disjunctions that are not very concise. This is which is often known by the "*replicated subtree*" problem [137].

Rules should be pruned by greedily removing conjuncts in the rule antecedent or even some rules. Let consider a conjunctive rule $r : A \rightarrow c_j$ where A is the rule antecedent and c_j is its consequent (class). In the pruning step, we look for many simplified versions $r' : A' \rightarrow c_j$ of the rule r. Then the *pessimistic error rate* is computed for each of these simplified rules, then the rule with the best quality measure is chosen to replace the original rule r. This process is iterated until the quality measure (The *minimum description length* principle) of the rule cannot be more improved. Rule pruning may produce identical rules, the next step consists to remove all duplicate rules.

Rule ordering

In contrary to the original rule set, pruned rules may lead to the multiple classification problem in which more than one rule can cover a given example reflecting a competition or a conflict situation.

To resolve the conflict between rules, C4.5 sorts rules by training (validation) accuracy to create an ordered decision list. To do this, C4.5 groups all rules that

predict the same class together into the same subset and then orders these class rule sets so as to minimize the number of false-positive errors (i.e., where a rule predicts a class c_j, but the real class is not c_j). The class that has the smallest number of false positives is given the highest priority since it is expected to contain the best set of rules. For the classification purpose, the first rule in the decision list that applies, is used to classify a test example.

Since pruned rules are not necessarily exhaustive, a default class should be defined in order the rule set covers the whole set of examples. A common way to choose the default class is to assign an example not covered by any rule to the largest or majority class. Other more sophisticated methods are developed in the related work of Chapter 4. In contrary to these methods, C4.5 does not choose the majority class, because this class will likely have many rules and many covered examples for each rule. Instead, it selects the class that includes the most training examples that were not covered by any rule.

Example 3.2.

Let us continue with the previous example treating the problem of classifying 'iris' plants. Using the original 150 examples from the 'iris' data set [131], C4.5 builds the decision tree given in Figure 3.1 in which, the partitioning of the data space by the decision tree is shown:

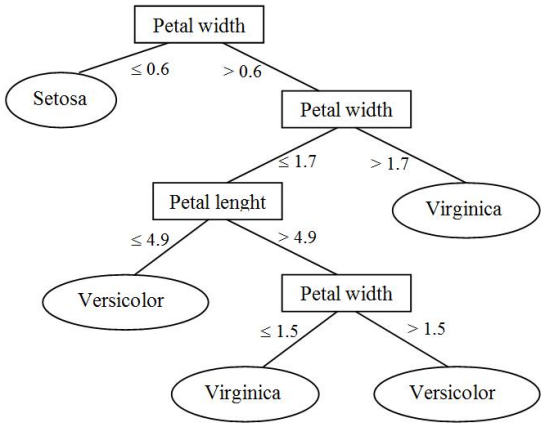

Figure 3.1: A decision tree constructed by C4.5 for the iris dataset [131]

The decision tree can be interpreted as follows: an iris plant with "petal width" less than 0.6 is classified as 'setosa' and an iris plant with a 'petal width' bounded in [0.6, 1.7] and a 'petal length' less than 4.9 is categorized as 'versicolor'. Three other remaining paths can also be drawn. Note that only 'petal width' and 'petal length' are used to construct the decision tree and the two remaining attributes are irrelevant. The Induced decision tree made a classification error of 6 out of 150 (total number of instances in the original data set, that is 4%. Therefore, it can be said that the error rate for this tree is 4%.

The decision tree of Figure 3.1 can be transformed to a set of classification rules by converting the path from the root node to each leaf to a classification rule. Extracted rules are in Figure 3.2:

r_1: If (petalwidth $\leq$ 0.6) $\rightarrow$ 'Iris-setosa '
r_2: If (petalwidth > 0.6) $\wedge$ (petalwidth $\leq$ 1.7) $\wedge$ (petallength $\leq$ 4.9) $\rightarrow$ ' Iris-versicolor'
r_3: If (petalwidth > 0.6) $\wedge$ (petalwidth $\leq$ 1.7) $\wedge$ (petallength > 4.9) $\wedge$ (petalwidth $\leq$ 1.5) $\rightarrow$ 'Iris-virginica '
r_5: If (petalwidth > 0.6) $\wedge$ (petalwidth $\leq$1.7) $\wedge$ (petallength > 4.9) $\wedge$ (petalwidth > 1.5) $\rightarrow$ ' Iris-versicolor '
r_6: If (petalwidth > 0.6) $\wedge$ (petalwidth > 1.7) $\rightarrow$ ' Iris-virginica '

Figure 3.2: Rules extracted from C4.5 decision tree for the iris dataset [131]

Let consider the two rules from Figure 3.2 predicting the class $'Iris-versicolor'$:

$r_2 : If(petalwidth > 0.6) \wedge (petalwidth \leq 1.7) \wedge (petallength \leq 4.9) \rightarrow' Iris - versicolor'$

$r_5 : If(petalwidth > 0.6) \wedge (petalwidth \leq 1.7) \wedge (petallength > 4.9) \wedge (petalwidth > 1.5) \rightarrow' Iris - versicolor'$

Observe that the first test $(petalwidth > 0.6)$ in r_2 is redundant if we assume that rules are ordered. In a decision list, a test example is classified by a particular rule only if all previous rules are not satisfied, thus this conjunct should be simplified considering that the rule r_1 is prioritized. Furthermore, if we observe that the rule set always predicts $'Iris - versicolor'$ when the value of *petallength* is less than 4.9, we can assume that the rule r_5 is a particular case of r_2 thus r_5 may be removed and we obtain only one rule for the class $'Iris - versicolor'$ as follow:

$$r'_2 : If(petalwidth \leq 1.7) \wedge (petallength \leq 4.9) \rightarrow' Iris - versicolor'$$

The same process can be applied to simplify rules for other classes.

3.4 Rule induction using Sequential Covering Algorithm

A common way to learn a rule based classifier is to extract IF-THEN rules directly from the training data without having to generate a decision tree first. The most

used rule induction methods exploit the *Sequential Covering Algorithm*. The basic idea of this algorithm is to sequentially learn rules (one at a time), where each rule should *cover* the maximum examples in a given class and hopefully none of examples of other classes.

Panoply of sequential covering algorithms has been developed in the literature. The most known include the AQ [134], CN2 [37] [36], FOIL [147] [149], GOLEM [135] and the more recent, RIPPER [39]. In the following, we will first give the general principle of sequential covering algorithm on which are based most of rule based learning methods before going through introducing the basis of each of these algorithms,

It should be mentioned that the general principle of sequential covering algorithm can be applied to induction of propositional rules as well as first order logic rules.

Mitchell (1997)[134] presents a sequential covering algorithm very cited in the literature. However, the pseudo-code of this algorithm by Mitchell is directly applicable only for induction of propositional rules in a downward way. Fürnkranz (1999)[89] presents an alternate version of this algorithm which can be applied to produce propositional rules as well as first order logic rules. Moreover, the algorithm enables to support various search schemes such as searching candidate rules in a descending or ascending way. The work by Fürnkranz also summarizes how the proposed algorithm can be instanced to be used in systems based on propositional rule induction techniques such as AQ by Michalski and CN2 or those for first order rule induction techniques such as FOIL and GOLEM.

3.4.1 General procedure of Sequential Covering Algorithm

The general strategy used in the sequential covering algorithm is as follows. In each iteration one rule is learned, examples covered by this rule are removed, and the process is iterated on the remaining examples. This sequential learning of rules differs from the heuristic used in decision tree induction. Since the path from the root to each leaf in a DT corresponds to a rule, we can say that DT model is a process of learning a set of rules simultaneously. Besides, to decide for the best branching, a DT evaluates the average quality of a number of disjointed sets (one for each value of the attribute that is tested), while rule based classifier only evaluates the quality of the set of instances that is covered by the candidate rule [114]

More formally rule-based learning using sequential covering algorithm can be summarized by the following steps:

1. To learn a set of rules for a class c_j, divide the training data into positive examples those that belong to c_j, and negative examples those belonging to

other classes;

2. Learn a conjunctive rule that covers as many positive examples as possible and only few negative examples;

3. Remove positive examples that are covered by this rule from the training set;

4. Repeat steps 2 and 3 as long as there are positive examples that are not covered by the generated rules or until a stopping criteria is met.

5. Repeat the above process for all other classes and learn a rule set for each of them.

The basic sequential covering algorithm is shown by the generic pseudo-code of Algorithm 4 proposed by Fürnkranz (1999)[89] (the SeparateAndConquer procedure).

Algorithm 4 Procedure SeparateAndConquer (Examples)

```
Theory = ∅
while POSITIVE(Examples) ≠ ∅ do
  Rule=FindBestRule(Examples)
  Covered = Cover(Rule, Examples)
  if RuleStoppingCriterion(Theory, Rule, Examples) then
    exit While
  end if
  Examples=Examples\ Covered
  Theory=Theory ∪ Rule
  Theory = PostProcess(Theory)
end while
Return (Theory)
```

As mentioned above, rules are learned for one class at a time. Ideally, we would like that each learned rule for a class c_j covers all (or at least many) of the training examples of this class and none (or few) examples from the remaining classes. This helps inducing rules of high accuracy but not necessary of high coverage. The whole coverage of all positive examples in this class is guaranteed because more than one rule can be learned for a specific class; these different rules may cover different examples for the same class. The process will iterate until a stopping criteria is met, such as when all training examples are covered by at least one rule or if the quality of an induced rule is below a given threshold.

In this algorithm, to learn a single conjunctive rule, a general- to-specific search can be performed which corresponds to a *beam* search. In this search mechanism, we start with a rule with an empty antecedent and then gradually add attribute tests

Algorithm 5 Procedure FindBestRule(Examples)

```
InitRule=InitializeRule(Examples)
InitVal=EvaluateRule(InitRule)
BestRule=(InitVal, InitRule)
Rules={BestRule}
while Rules ≠ ∅ do
  Candidates=SelectCandidates(Rules, Examples)
  Rules=Rules \ Candidates
  for Candidate ∈ Candidates do
    Refinements=RefineRule (Candidate, Examples)
    for Refinement ∈ Refinements do
      Evaluation=EvaluateRule(Refinement, Examples)
      unless StoppingCriterion(Refinement, Evaluation, Examples)
      New Rule=(Evaluation, Refinement)
      Rules=InsertSort(NewRule, Rules)
      if NewRule > BestRule then
        BestRule=NewRule
      end if
    end for
  end for
  Rules=FilterRules(Rules, Examples)
end while
Return (BestRule)
```

to it that optimizes an objective function until it covers only positive examples or a stopping criterion is checked.

After a rule is learned, post-pruning is usually conducted to prevent the rule from overfitting the data. In post-pruning, some of attribute tests in the rule are removed if their removal improve or at least do not decrease the classification efficiency of this rule according to a quality measure.

The *FindBestRule* procedure in Algorithm 5 finds the best rule for the current class, given the current set of training examples. This procedure includes the following operations [121]:

- An initial rule (*InitRule*) is used to initialize the search process which could be very general (with an empty antecedent), or very specific (directly represented by an example) or a combination of the two cases.
- The initial rule is then evaluated (*EvaluateRule*) using some criteria (e.g. information gain, other criteria are detailed in the following section). This rule is assumed to be the best one and is added to the list of rules to be considered.
- The procedure performs a set of iterations, in each of them a sub-set of candidate rules is selected using *SelectCandidates*; and removed from the list of rules to be considered. This procedure choose the subset of rules to be generalized/specialized. It involves a random search for the number of best rules considered to be refined.
- Candidate rules are then refined (*RefineRule*), producing a set of refinements (every technique use a specific refinement process). Possible refinements are evaluated using (*EvaluateRule*) and may be filtered by some criteria and then added to the list of rules to be explored. The refinement having the best evaluation will replace the current best rule.
- The current rule can optionally be filtered according to a specific search method; one or more rules can be refined during the iteration. This procedure can use the same search method as of the *SelectCandidates* procedure.

This general pseudo-code recapitulates, with a very concise manner, the general structure of a panoply of rule induction algorithms.

3.4.2 Principal choices in sequential covering method

The majority of rule induction algorithms based on sequential covering method differ by at least these four points: (1) the rule representation, (2) the search mechanism

used to explore the candidate rules space, (3) the rule evaluation measures and (4) the pruning methods. The items (2) (3) and (4) will be discussed in the following with more details.

The search mechanism

The search mechanism can be divided to a search strategy and a search method. The search strategies refer to the global directions of search: $top - down$, $bottom - up$, or *bidirectional* search:

- A $top - down$ search starts from the most general rule hypothesis (a rule with an empty antecedent) and then gradually specify it. This search type is the most used in rule learning (it is used for example in AQ, CN2 and FOIL).
- A $bottom - up$ search consists to start with one or more specific rule hypothesis and then make them more generic (by pruning). This type of search is only supported by very few rule learning techniques (such as the system GOLEM). A common problem involved in ascending search is the choice of the starting rule hypothesis. Given a set of training examples, a large number of initial hypotheses is possible. In contrary, descending search trivially determines the starting rule hypothesis by generally choosing the most generic rule (something true unconditionally).
- *Bidirectional search* combines both descending and ascending search. There exist only little of techniques supporting this search strategy.

The search method corresponds to the exploration techniques of the search space. Fürnkranz (1999) [89] described the search algorithms of mainly four different types: $hill - climbing$, *beam search*, $best - first\ search$ and *stochastic search*. In the general algorithm presented in Section 3.4.1, these search techniques can be considered in procedure $FindBestRule$ in a various ways, including the iteration on rule candidates, the sub-procedure $RefineRule$, and $FilterRules$:

- The $hill-climbing$ search is usually the most used technique which is supported for example by the algorithms AQ, $CN2$, $FOIL$ and $GOLEM$. This search consists in selecting the rule hypothesis having the best evaluation in a given space and continuing this search by refining this rule recursively. Although, the $hill - climbing$ search is limited to the local maxima, it is usually efficient.
- The *beam* search is also rather usually used and supported by the AQ and CN2 algorithms. This type of search adds to the $hill - climbing$ search the possibility of exploring a number N (one or more) of better rule hypothesis. The

number N corresponds to the width of the search beam: value 1 corresponds to the $hill - climbing$ search. The main advantage of the *beam search* is its ability to reduce (without generally eliminating it) the myopia relative to the hill-climbing search, which is limited to the local minima. The beam search is however more expensive in terms of search time.

- The $best - first$ search selects the best candidate hypothesis and all its refinements without using any pruning criteria (sub-procedure $StoppingCriteria$ in Algorithm 5). Starting from the basic general hypothesis and without any pruning, the best-first search can find the optimal rule. This type of search is however always very expensive in terms of time and is used only in relatively few rule-based learning techniques.
- The *stochastic* search usually introduces a certain degree of randomness into the rule refinement (sub-procedure $RefineRule$ in Algorithms 5); this aims to avoid the local minima while also avoiding expensive search. Stochastic search is only used in few rule-based learning techniques.

Pruning methods

Each rule induction method, has its particular procedure to learn simple and specific rules. Almost all of these methods adopt a pruning heuristic which helps to avoid overfitting and trait noisy data. Pruning can occur during rule learning (pre-pruning) or in a posterior step (post-pruning). The pre-pruning enables to stop rule refinement before theses rules become too specific or before reaching overfitting. Among methods for rule pre-pruning, we can cite the following heuristics. The *statistical significance test* is a criterion used to stop rule generalisation/specification. It compares the distribution of the observed class, among examples covered by the rule, with the predicted distribution resulting if the rule has randomly chosen examples. Other pruning criteria are also used for example, the *minimal purity* which requires a positive percentage of covered examples by a rule should be maintained. The *restriction of coding length* used in FOIL, aims to restrict the length of the rule in order to avoid rules with complex antecedent usually covering few examples. The *cut* criteria can also be used [89].

Post-pruning helps to improve the induced model after construction. The main idea is to remove some rules or conjunctions in each rule if this removing preserve or improve the predictive accuracy of the rule classifier. The most used post-pruning techniques are *reduced pruning error* (REP)[29] and *Grow* [38].

Pre-pruning based learning techniques are more efficient (in terms of rapidity) than post-pruning based learning techniques. However, post-pruning usually extracts models with high accuracy and with a more simple rules than that of the pre-pruning.

Intuitively, this related to the fact that post-pruning has more available information (the hole model is induced) to make decision and so tries to be less specific than pre-pruning. We note that the majority of post-pruning techniques, removing one rule conjunction in each time, are still greedy.

Rule quality measures

The procedure *FindBestRule* needs a rule quality measure to decide which rule should be retained. At each iteration a test on an attribute should be checked in order to see if adding such a test to the rule antecedent will improve the rule or not. The heuristics for rule quality measure are implemented in the sub-procedure *EvaluateRule*. One can simply use *accuracy* for this evaluation, but this isn't always the best quality measure. Let consider the following example:

Example 3.3.

Let us consider that we have two rules r_1 and r_2, both are for the *positive* class (+). Rule r_1 correctly classifies 40 of the 43 examples it covers. Rule r_2 covers only four examples, which it correctly classifies. The accuracies for r_1 and r_2 are thus respectively 93.02% and 100%. It seems that r_2 is preferred to r_1 because it has higher accuracy. However, r_1 is more representative of the dataset since its covers more examples than r_2. From the above example, we can say that both accuracy and coverage taken alone cannot be a good quality measure: for a given class we could have a rule that covers many examples, most of them belong to other classes. On the other hand, rules with high accuracy may cover only few examples.

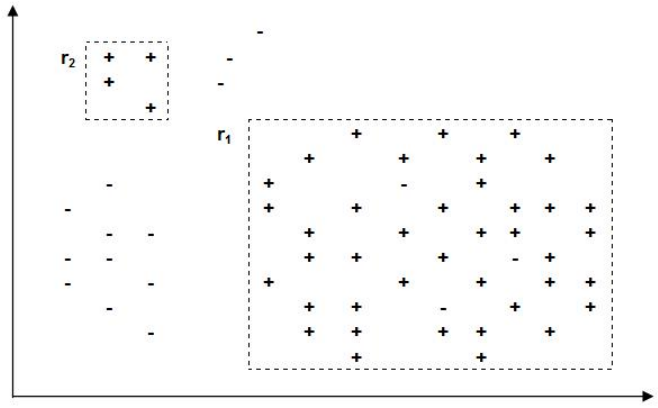

Figure 3.3: Rules for the positive class including positive and negative examples

For this reason some other rule quality measures, which combine both accuracy and coverage, have been developed. In the following, we only present the most used evaluation criteria, denoted *Statistical test*, *Laplace measure* and *Information Gain*.

A more complete survey of most used rule quality measures can be found in [2] which also includes a comparative study between these approaches.

- **A statistical test of significance** can be used to see if the correct classification made by a rule is not attributed to random guessing but instead reflects a real correlation between attribute values and classes. In this test, the observed frequency among classes of examples that are covered by the rule are compared to the expected frequency obtained if the rule randomly made predictions.

  The following likelihood ratio statistic is computed:

$$Likelihood\_Ratio = 2\sum_{j=1}^{C} f_j log(\frac{f_j}{e_j}) \tag{3.8}$$

  where C is the number of class labels, f_j is the observed frequency of class c_j i.e. examples that are covered by the rule, and e_j is the expected frequency of a rule that makes random predictions. The statistic has a chi-square distribution with C-1 degrees of freedom. A large ratio shows that the number of correct predictions made by the rule is $significantly$ larger than that expected by "random guesser".

- **Laplace estimate**: this metric aims to penalize rule having low coverage, it favours rules covering few negative examples but many positive examples over a rule that do not covers any negative examples but only few positive examples. This evaluation measure is used in recent versions of CN2.

$$Laplace = \frac{p+1}{n_{cover}+C} \tag{3.9}$$

$$m - estimate = \frac{p + CProb^{+}}{n_{cover}+C} \tag{3.10}$$

  where n_{cover} is the number of examples covered by the rule (positives and negatives), p is the number of positive examples covered by the rule, C is the total number of classes, and $Prob^{+}$ is the prior probability for the positive class. Note that the m-estimate is equivalent to the Laplace measure by choosing $Prob^{+} = 1/C$.

  These measures are largely affected by the rule coverage. If the rule has large coverage, then both measures tends to the rule accuracy p/n_{cover}, however if the rule coverage is close to 0, then the Laplace measure is reduced to $1/C$, which is the prior probability of the positive class assuming a uniform class distribution. So we can say that these two metrics reflect a balance between accuracy and prior probability of the positive class.

- **The FOIL's Information Gain**: Another measure is based on information gain which is firstly proposed in the algorithm FOIL. Let us consider the rule $r : A \rightarrow +$ which covers p positive examples and n negative examples. We would like to know if an added conjunct B to the antecedent of r resulting a new rule $r^{'} : A \wedge B \rightarrow +$ could contribute to improve the rule quality. We assume that the new rule $r^{'}$ covers $p^{'}$ positive examples and $n^{'}$ negative examples.

  For the positive class (+), the amount of information necessary to correctly classify an instance into class (+) whose prior probability is Prob(+) is defined by [113] $-log(Prob(+))$ [bit], where the log function is of base 2. If we consider the rule r, the amount of information we need to correctly classify an instance into class (+) is $-logProb(+ \mid r)$ [bit], where $Prob(+ \mid r)$ is the posterior probability of the class (+) given r. Now taking into account the two rule versions (r and $r^{'}$), the amount of information obtained by adding the new conjunct B is: $logProb(+ \mid r^{'}) - logProb(+ \mid r)[bit]$. By using frequencies to estimate probabilities, the Foil-Gain is defined by:

$$FOIL\_Gain = p^{'} * (log_2 \frac{p^{'}}{p^{'} + n^{'}} - log_2 \frac{p}{p+n}) \tag{3.11}$$

  Since this measure is proportional to $p^{'}$ and $\frac{p^{'}}{p^{'}+n^{'}}$, rules having high accuracy and coverage will be preferred.

3.4.3 Rule induction algorithms

In this section, we present some sequential covering based algorithms. We first give the background for the well known RIPPER algorithm. Then, we only introduce a brief descriptions of the CN2, FOIL and GOLEM algorithms. For more details about algorithm CN2 see [37][36], and for learning first order rules from propositional ones see [116].

The RIPPER algorithm

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) is a well-known rule-based algorithm initially proposed by Cohen [39]. It is the successor of IREP algorithm for rule induction [90]. Although the two algorithms have the same principal, the RIPPER improves IREP in many details and is also able to cope with multiclass problems.

This algorithm scales almost linearly with the number of training examples and is especially appropriate for inducing rules from data sets with imbalanced class

distributions. RIPPER is also efficient to deal with noisy data sets because it uses a validation set to avoid overfitting. For two-class problems, RIPPER use the majority class as a default class and learns rules representing the minority class.

For multiclass problems before learning rules, the training data is sorted by class labels in ascending order according to the corresponding class frequencies. Let $(c_1, c_2, ..., c_C)$ be the ordered classes, where c_1 is the least frequent class and c_C is the most frequent class. Rules are then learned for the first C-1 classes, starting with the least frequent ones. During the first step, instances that belong to c_1 are labeled as positive examples, while those that belong to other classes are labeled as negative examples. The sequential covering method is used to generate rules that discriminate between the positive and negative examples. Next, RIPPER extracts rules that distinguish c_2 from other remaining classes. This process is repeated until it finds no more rules to learn. At this moment, a default rule (with empty antecedent) is added for the last (the most frequent) class.

The RIPPER algorithm is composed of two principal phases: the rule learning and the rule optimization. In the first phase an initial rule base is built using the rule induction algorithm denoted IREP* [90]; in this phase two steps are involved: the rule growing and rule punning.

The rule growing: The initial form of a rule is just a conclusion (the class value) and an empty antecedent. In each iteration the best conjunct, based on its $Information\ Gain$, is added to the antecedent. The process of growing conjuncts in the rule antecedent is stopped if we reach to an empty set of non covered positive examples or if no improvement in the information gain score can be observed. The information gain measure is calculated as in Equation 3.11.

The rule pruning: Pruning is an attempt to prevent rules to be too specific. Pruning is made according to a metric denoted $V^*(R)$. IREP* chooses a literal (conjunct) candidate to be pruned based on a score which will be computed for all rule antecedent literals when applied to the validation data. This score is defined as follows:

$$V^*(R) = \frac{p - n}{p + n} \tag{3.12}$$

where p (respectively n) is the number of positive (negative) examples in the validation set covered by the rule. This metric is especially related to the rule's accuracy when confronted to the validation set.

A conjunct is removed if the above evaluation metric will be improved after

pruning. The pruning is involved starting from the last conjunct added to the rule. For example, let $r : a_1 \wedge a_2 \wedge a_3 \wedge a_4 \rightarrow c_j$, RIPPER checks if a_4 should be pruned first, followed by $a_3 \wedge a_4$, $a_2 \wedge a_3 \wedge a_4$, etc. To ovoid overfitting, the pruned rule may cover some of negative examples and is more efficient for generalization than the original rule if confronted to unseen examples.

The last phase in this algorithm consists to optimize the already generated rule base. Rule optimization involves a process of *replacement* or *revision* of each rule in the rule set which aims to determine whether some of existing rules can be replaced by better alternative rules. To decide for the best version of each rule to retain, the algorithm uses the MDL (*Minimum Description Length*) criterion [148].

By applying it to several databases, this algorithm is simple and generates more robust rules than those generated by the C4.5 algorithm in the rule mode [76].

Example 3.4.

This example presents the rule set generated for the 'Iris' data set using the RIPPER algorithm. The induced model, given in Figure 3.4, includes four rules for different class labels where the last one is a default class.

r_1: If (petallength $\leq$ 1.9) $\rightarrow$ ' Iris-setosa '
r_2: If (petalwidth $\geq$ 1.7) $\rightarrow$ ' Iris- virginica'
r_3: If (petallength $\geq$ 5) $\rightarrow$ 'Iris-virginica'
r_4: $\rightarrow$ ' Iris-versicolor '

Figure 3.4: A rule set generated by the RIPPER algorithm for the iris dataset [131]

The CN2 algorithm

CN2 [37][36][134] is an algorithm for induction of propositional rule. It integrates functionalities of two types of algorithms; it integrates the basic principal of AQ algorithm using a $beam - search$ exploration of data; and it introduces the ability of ID3 algorithm [37] for decision trees induction to cope with noisy data, especially the use of statistical methods for rule evaluation.

The first versions of CN2 [37] use the entropy in the rule evaluation whereas the more recent versions [36] use Laplace estimate, which allows better results when the data are noisy.

Also we note that the oldest versions of CN2 produce ordered rules whereas the more recent versions enable to produce unordered rules. CN2 uses a significance

test as an heuristic for pre-pruning. The sequential covering algorithm presented by Mitchell (1997)[134] is well applied to the CN2 algorithm. Inside the procedure of search for the best rule, attributes are iteratively added to the candidate rule antecedent in order to specialize them until it reaches a suitable covering level. The CN2 algorithm induces rules for two class problems where a default rule is predicted for the negative class.

The FOIL algorithm

FOIL [147][149][134] is an algorithm for learning Horn clauses. As in the case of CN2 algorithm, FOIL produces rules in a downward way and uses statistical metrics for evaluating and pre-pruning rules. However, statistical methods used by FOIL are different from those used by CN2: for rule evaluation, FOIL uses metrics based on a weighted information gain but adapted to literals and variables; for pre-pruning FOIL uses metrics based on the *Minimum Description Length* (MDL).

FOIL only supports $hill - climbing$ exploration whereas CN2 supports both $hill - climbing$ and $beam - search$ exploration at the same time. In addition, FOIL searches for rules predicting only positives concepts. The rule-based learning enables producing rules whose conclusion is a predicate. In the rule refinement step, FOIL recursively adds premises to the rule in the form of literals. Each literal is an existing predicate, an equality function, or a negation of this predicate or equality function. Some used heuristics enable introducing a new variable which is not related neither to the conclusion nor to literals of the current premises.

The GOLEM algorithm

GOLEM is also an algorithm for learning Horn clauses [135]. Contrary to CN2 and FOIL, GOLEM applies a rule search strategy in an ascending way. GOLEM starts the search process by a too specific rule formed from a pair of random positive examples, then it generalizes this rule by a hill-climbing exploration. The technique of ascending exploration used by GOLEM is based on the reversed resolution and the procedure of *relative least general generalization*: this procedure combines two or more clauses in only one clause and can apply a substitution operation to the variables.

3.5 The PART Algorithm

To summarize the two previously presented sections, we can say that rule induction for classification purpose operates in two-steps. First, a set of rules is initially induced

which should be representative of the training data. Then, this rule set will be refined later according to a global optimization procedure. This process can be performed by two different ways: either by using the *Sequential Covering* algorithm including a rule refinement strategy able to produce a set of optimized rules or by constructing a decision tree, transforming the tree to a rule set and then pruning the rule set as already described.

It has been shown in [85] that rules induced by $separate-and-conquer$ methods may have over-pruning usually called *"hasty generalization"*, while rules extracted from a decision trees are computationally expensive especially in presence of noisy data [39]. As an attempt to avoid problems encountered with these two techniques, Frank and Witten (1998)[85] proposed an algorithm called PART (Partial decision trees) which is basically founded on combining the two previous approaches.

PART induces the set of rules as follows. The main separate and conquer idea is maintained where rules are iteratively built and all their covered examples are removed until all examples are covered. However, it differs from the standard approach in the way that each rule is learned. In particular, to learn one rule a partial pruned decision tree is built for the current set of examples, the leaf with the largest coverage is made into a rule, and the tree is discarded. The pruned decision tree helps to avoid the over-pruning problem of methods that immediately prune an individual rule after construction. In addition, inducing partial decision trees can avoid expensive rule optimization procedure performed in decision tree based rule learning.

The authors in [85] have shown that this algorithm leads to less complex rule size and higher accuracy if compared to both C4.5 decision tree [145] and $separate-and-conquer$ based RIPPER [39]. See [85] for a comparative study between RIPPER, C4.5 and PART.

Example 3.5.

This example shows the rule set generated for the "iris" data set using the PART algorithm which includes three rules. The last induced rule is the default class.

r_1: If (petalwidth ≤ 0.6) $\rightarrow$ 'Iris-setosa '
r_2: If (petalwidth ≤ 1.7) $\wedge$ (petallength ≤ 4.9) $\rightarrow$ ' Iris-versicolor'
r_3: $\rightarrow$ ' Iris-virginica '

Figure 3.5: A rule set generated by the PART algorithm for the iris dataset [131]

Although the two decision lists obtained by RIPPER and PART algorithms have the same accuracy (94%), obtained by applying the WEKA software implementation of the two algorithms [168], the rule set generated by PART seems to be more compact (See Figures 3.4 and 3.5).

3.6 Conclusion

In this chapter, we reviewed principal rule based learning methods. In particular, we focused on two families of rule based classifiers: the first family learns a classical decision tree and then extracts IF-THEN rules using a rule extraction process. The second family is rather based on the sequential covering algorithm which enables extracting rules directly from data without going through a decision tree model. Positive and negative aspects of these two family of methods are also discussed. In the last part of the chapter, we described a particular rule based classifier; the PART algorithm; which combines the sequential covering principle and the decision tree extraction in a same algorithm in order to overcome some drawbacks of these methods. In the next Chapter, the PART algorithm will be used as a basic algorithm to propose a variety of fuzzy versions of this classifier in the hope to improve its efficiency.

Chapter 4

A Possibilistic Rule based Classifier

4.1 Introduction

In many applications, rule induction algorithms gain the most popularity among machine learning techniques. If the number of rules is relatively small and accuracy is sufficiently high such classifiers are a good choice. Rules are desirable since they are more readable [168] than black box methods such as SVMs. However searching for a set of rules that optimize the cost function based on the error rate made by the classifier is one the main issues to tackle in rule induction methods because this searching is considered as a difficult optimization problem [76] [142] [155].

As previously introduced in Chapter3, once rules have been induced from training instances, there are three possible situations that demand different solutions, when classifying unseen examples [125]:

- One-covering case: one or more rules cover a test example and are associated with the same class. In this easy case, the example is assigned to this class.

- Multiple covering case: several rules cover the example and they are associated with different classes. This is clearly an issue because, even if one of the rules correctly classifies the example, it is considered as wrongly classified since no rule can be preferred in such situation. A default choice of the rule may deteriorate the classification accuracy.

- Non-covering case: no rule covers the example. As in the multiple covering case, the example is considered as wrongly classified. In this case, a simple solution is to take the most frequent class as a default choice. One might also associate the class of the closest training example.

In this chapter, we propose a family of Possibilistic Rule-based Classifiers (PRC) to deal with such problems which are an extension and a modification of the PART algorithm [85] previously described in Chapter 3.

The PRCs, which keep the rule learning process of the PART algorithm, differ from it in more than one respect. First, the PRCs generate a set of fuzzy rules instead of crisp rules. Fuzzy rules, by their flexible boundaries, enables a gradual transition when discretizing real-valued attributes, and so overcome the drawbacks of crisp rules making an abrupt transition [97]. The rationale behind the use of fuzzy sets is that they make a distinction between rules that are strongly satisfied when the attribute value (of the example to classify) is inside the core of the fuzzy sets and rules that are weakly satisfied when this value is near the boundaries.

Second, in the classification phase, PRCs evaluate rules in an *unordered* manner instead of using a decision list, i.e. all rules in the rule set are evaluated in an *equivalent* way by estimating their relevance and then choose the rule with the highest estimate. The main advantage of using unordered fuzzy rules, compared to decision lists, is that the order among rules is determined by their degree of satisfaction for the current example. Finally, fuzzy rules are more flexible, cover larger sets of cases than crisp rules and contribute to reduce (or totally eliminate by using very large supports) the number of non-covered examples.

This chapter is structured as follows: Section 2 reviews some related works. We describe along Section 3, the architecture of the Possibilistic Rule based Classification approach where we present the rule fuzzification algorithm and the Possibilistic Rule-based Reasoning. The hybrid version is detailed in Section 4. In Section 5, we report the experiments for all PRCs that we propose in this chapter. Finally, Section 6 concludes and proposes some directions of future research.

4.2 Related works

Getting a set of rules that does not lead to ambiguity in classification for the training set, does not guarantee that multiple (or the absence of) classification will not occur when considering a new example. Many methods have been proposed to deal with this problem. The simplest one is to use decision lists [39] [79] where a test example is classified using the first satisfied rule. Serrurier and Prade [155] have proposed a formalization of the ILP problem using first-order possibilistic logic to deal with multiple classification. In order to prevent an example to be classified in more than one class, the authors exploit the fact that possibilistic logic associates a priority level to each rule. However in this kind of method, rules need to be sorted at the induction step, which may favor some rules and penalize others in a non symmetric way.

In contrast with decision lists, some other methods learn rules in an unordered manner. To deal with the multiple classification problem, a first idea is to assign weights to the rules at the deduction step in order to distinguish between the satisfied rules. One type of approach for computing these weights is to learn fuzzy rules instead of crisp ones. Thanks to their flexible boundaries, fuzzy rules in numerical settings no longer cover an instance with a $\{0, 1\}$ degree, but rather in a gradual manner using a membership function with values in [0,1]. This process enables us to distinguish between *strongly* satisfied rules and those *weakly* satisfied for a given example.

Ishibuchi and Yamamoto [101] have proposed heuristic methods to define weights for fuzzy rules. Mainly two strategies are adopted: i) the single winner method: where the rule having the highest estimate, in terms of its compatibility grade and its certainty factor (weight), is used to classify the example and ii) the weighted vote method: where the vote is for the class with the highest estimate defined by the sum of products of the compatibility grade and the certainty factor of rules labeled with this class (see [101] for details).

In the same context, the estimate of probability (EP) method [133][125] used in the HCV induction algorithm assigns an EP value for each class by examining training set coverage of satisfied rules for this class. More recently the FURIA algorithm [97] which is an extension of the RIPPER algorithm [39] has been proposed. In this work, the authors use a rule fuzzification algorithm for learning unordered rules instead of rule lists and adopt a weighted vote method, whereas the classifiers that we are going to present use the single winner method. Still FURIA looks similar to our PRC since it is a fuzzy extension of RIPPER algorithm as the PRCs are an extension of the PART algorithm. However, the main difference is that FURIA uses only one type of fuzzification, whereas our PRCs exploit different forms of fuzzification to deal with different situations (see Section 3).

A standard way to deal with the non-covering problem is to adopt a default class strategy, which assigns an example not covered by any rule to the most frequent class [9][37]. The choice of the default class is done at the induction time and is not dynamically updated at the classification time [125]. Using a pre-fixed default class for any example in the test set could deteriorate the classification results.

An alternative to a fixed default class strategy, is to consider a neighborhood of the unclassified example in the training set [133][125]. The Measure of Fit (MF) method [133][125] calculates the MF value of each class for each test example not covered by any rule. The MF value is interpreted as the closeness of the test example to a class based principally on the rule coverage of the training set for that particular class.

In [22] the authors proposed a possibilistic approach which *dynamically* assigns

a default class to each example not covered by any rule. For this purpose, they used a possibilistic class based-reasoning which searches for the most plausible class by quantifying relationship between examples and classes through a distance measure.

Other methods have been proposed in this context, for example the FURIA algorithm [97] makes use of a stretching method which generalizes induced rules until they cover an example initially not covered by any rule. The rule generalization method is first proposed in [77]. It searches for the optimal rule antecedent to delete in order to cover the example. The authors evaluate rules using Laplace accuracy on the training data and classify the example by the rule with the highest evaluation. In the context of rule fuzzification, Muñoz et al. [127] have recently proposed an algorithm to extract fuzzy classification rules from training examples. This algorithm is a fuzzy extension of the well-known CN2 algorithm [37] which exploits linguistic hedges to obtain more precise and compact rules. Contrary to our proposed rule based classifier, this approach makes use of *ordered* rule lists.

4.3 The Possibilistic Rule-based Classifiers: PRCs

In this chapter, we propose a family of Possibilistic Rule based Classifiers using the architecture described in Figure 4.1. These classifiers use a training module which allows to learn a set of crisp rules from training examples based on the standard learning process used by the PART algorithm [85] (detailed in section 3.5 page 93). We recall that this latter is basically founded on combining the sequential covering principle [89] and the decision tree induction method as in C4.5 algorithm [145]. The PART algorithm is considered among the best-known rule induction algorithms in terms of simplicity of induced rules and accuracy [85].

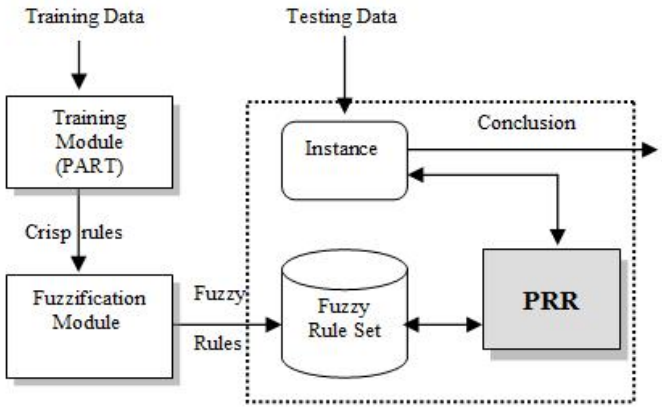

Figure 4.1: Architecture of the Hybrid Possibilistic Classification approach

The PRCs, which keeps the same rule learning process of the PART algorithm, aims to extend and modify this algorithm in order to improve its efficiency. Even if

the proposed classifier can be easily adjusted to classify certain or uncertain data, we are only interested to deal with certain numerical data in this work. The PRCs differ from the PART algorithm in more than one sight. First, the PRCs generates a set of fuzzy rules (through the fuzzification module) instead of crisp rules. Fuzzy rules, by their flexible boundaries, enables a gradual transition between classes and so overcome drawbacks of crisp rules making an abrupt transition [97]. The intuitive behind the use of fuzzy rules is that they contribute to distinguish more between rules that are strongly satisfied when the attribute value (of the example to classify) is inside the core and rules that are weekly satisfied when this value is near to the boundary.

The main contribution of PRCs over the standard PART is in the classification phase. In fact, the classification module is founded on a Possibilistic Rule-based Reasoning (PRR in Figure 4.1) which uses the generated fuzzy rules to classify testing instances. The PRR allows sorting rules according to their degree of Possibilistic Relevance to each test example. The PRR (detailed in Section 4.3.2) evaluates rules in an unordered manner instead of using decision lists as in the case of the PART algorithm, i.e. all rules in the rule set are evaluated in a *symmetric* way by estimating their relevance possibility and then choose the rule with the highest estimator. In a decision list, an example is classified by the first satisfied rule in the list. These rules are sorted at the induction step according to their coverage for example and never adapted after in the deduction step. The main advantage of using unordered rules, if compared to a decision list, is sorting satisfied rules at the deduction step depending on their degree of satisfaction against the current example.

Finally, using a pre-fixed default class to classify all non covered examples (based on the largest class for example) seems to be disadvantageous since this class may be inappropriate to some examples. To deal with such problem, we propose to extend and so generalize fuzzy rules in order to cover examples not covered by any classical rule. Rule generalization enables to seek for an adjusted default class specific to each test example. In this context, we propose a rule fuzzification approach which searches for the *largest support* for rules that could extend crisp rules in order to cover all non covered examples.

Although, we are only interested to the PART algorithm in this work, the generic nature of the proposed architecture given in Figure 4.1 allows it to be a possibilistic extension for any other classical rule based classifier that learns crisp rules such as the RIPPER algorithm [39].

4.3.1 Rule fuzzification

In this section, we first present fuzzy rules, and we then propose a new algorithm for rule fuzzification which is used to refine crisp rules, learned by the PART algorithm,

into fuzzy rules.

Rule representation

In most rule based classifiers, a rule is of the form:

$$IF\langle antecedent\rangle THEN\langle consquence\rangle$$

where *antecedent* is the conjunction of a set of n selectors $\{A_1, ..., A_n\}$. When we deal with numerical attributes, the general syntax of a selector is:

$$\langle Selector\rangle ::= \langle LowerBound\rangle\langle\ Operator\rangle\langle Attribute - Name\rangle\ \langle Operator\rangle \langle UpperBound\rangle$$

with Operator $\in \{<, >, \leq, \geq\}$.

Note that the PART algorithm produce rules with selectors limited only in one part i.e. selectors of the form: $(A_i \leq u)$ or $(A_i \geq l)$. If we only consider normalized numerical attribute values in $[0, 1]$, this means that $A_i \in [0, u]$ or $A_i \in [l, 1]$.

A fuzzy rule is obtained by replacing each selector in the initial crisp rule by a fuzzy selector with trapezoid membership function. A fuzzy selector A_i^F is identified by the *core* and the *support* which can be for the upper (C_U and S_U) or the lower bound (C_L and S_L) as in Figure 4.2. Given an attribute value a_i and a fuzzy selector A_i^F with the same attribute name, a membership function $\mu_{A_i^F}$ reflects to what extend a_i is covered by A_i^F with values in $[0, 1]$ defined as follows:

$$\mu_{A_i^F}(a_i) = max(0, min(\frac{a_i - S_L}{C_L - S_L}, 1, \frac{S_U - a_i}{S_U - C_U})) \tag{4.1}$$

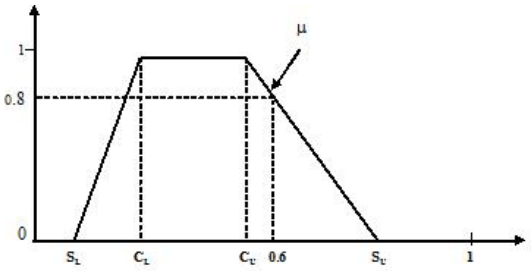

Figure 4.2: Trapezoid membership function

In this work, we propose three fuzzification approaches to deal with different aspects [26]:

Approach A: We consider a restrictive reading of the crisp rule. In this case the initial rule set is refined by a fuzzy rule set with the *same* support and a *restricted* core. We expect that this fuzzification approach will help to separate the multiple-classification cases.

Approach B: We consider a reading of the crisp rule that is robust to small variations of data. In this case, a selector in the original rule is refined by a fuzzy set having a *larger* support and a more *restricted* core. More precisely, for each fuzzy selector, the support and core should be chosen so that the 0.5-cut corresponds to the classical crisp selector. It may help to separate multiple-classification cases, while taking advantage of rule robustness.

Approach C: We consider a permissive reading of the crisp rule. A classical rule selector is then refined by a fuzzy selector with the *same* core and a *wider* support. This last approach contributes to decrease the number of non-classification cases. By using very large supports the non-covering problem may be totally solved.

Parts b, c and d in Figure 4.3 show the representation of fuzzy rules for each proposed approach where x_i is the value of the crisp selector in the rule, the grey part corresponds to the original set, and the dotted lines to its fuzzy version.

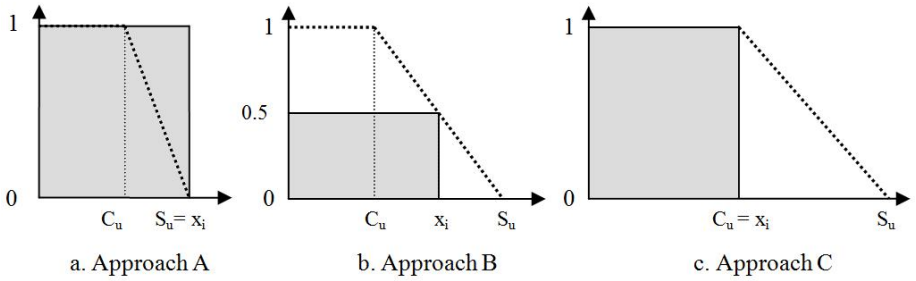

Figure 4.3: Fuzzification approaches

Rule Fuzzification Algorithm

In order to fuzzify a rule in the rule set, we propose to fuzzify each selector in this rule independently to other selectors as described in Algorithm 6 [26].

The process of selector fuzzification amounts to search for the best core (approach A or B) or support (approach B or C) that could extend a crisp selector to a fuzzy one. Given the training set T_r for each crisp selector A_i, we look for all possible attribute values in T_r that could be its core or support. Core (resp. support) candidates are those satisfying condition $C_L \in]x_i^L, 1]$ (resp. $S_L \in [0, x_i^L[$) if the crisp selector is of the form $(A_i \geq x_i^L)$ and $C_U \in [0, x_i^U[$ (resp. $S_U \in]x_i^U, 1]$) if selector is of the form

Algorithm 6 Fuzzify (r_j)

n = Number of selectors of r_j
$r_j^F = \emptyset$
for $i = 1$ to n **do**
  $A[i]^F = BestFuzzification(A[i])$
  $r_j^F = r_j^F$ AND $A[i]^F$
end for
Consequence (r_j^F)= Consequence (r_j)
return r_j^F

($A_i \leq x_i^u$) (Figure 5.2). Then we evaluate these candidates in terms of their accuracy and their proximity to the crisp value of the selector. The core or support used to fuzzify A_i is that having the highest *quality* (Equation 4.3).

Algorithm 7 describes the search process of support candidates. Analogous process is used to search for core candidates by reversing conditions. For approach B, the process returns to search for all possible cores and supports. For each of them, we make sure that the 0.5-cut corresponds to the original crisp version.

Algorithm 7 BestFuzzification(A_i)

T_r = Number of training Instances
Supports = $\emptyset$
$BestSupp = \emptyset$
$Quality_{max} = 0$
for $k = 1$ to $\mid T_r \mid$ **do**
  if ($a_i^k < x_i^L$) **then**
    Add a_i^k to Supports
  end if
  if ($a_i^k > x_i^U$) **then**
    Add a_i^k to Supports
  end if
end for
for each Support S **do**
  $Quality(S)$ = GetSelectorQuality(S)
  if $Quality(S) > Quality_{max}$ **then**
    $Quality_{max} = Quality(S)$
    BestSupp = S
  end if
end for
return BestSupport

To measure the quality of fuzzy selector, we use the *accuracy* defined by:

$$acc(A_i^F) = \frac{tp}{tp+fp}, tp = \sum_{a \in T_r^+} \mu_{A_i^F}(a), fp = \sum_{a \in T_r^-} \mu_{A_i^F}(a) \tag{4.2}$$

T_r^+: the training set labeled with the same class as the rule $r_j | A_i^F \in r_j$.

T_r^-: the training set labeled with other classes than that of r_j.

The accuracy measure favor selector with core/support very far from the value of crisp selector x_i which enables fuzzy sets to be too flatten and thus have more conflicting decision bound between classes. That's why, we investigate a more suitable quality measure for fuzzy selectors which combines the core/support accuracy (of the selector) with its proximity to the crisp value in a conjunctive manner:

$$Quality(A_i^F) = acc(A_i^F) * prox(A_i^F) \tag{4.3}$$

$$prox(A_i^F) = 1 - |distance(core(or support), x_i)| \tag{4.4}$$

This quality measure leads to choose a fuzzy selector with the nearest core/support to x_i among those having high accuracy. To compare our fuzzification algorithm to that proposed by [97] in terms of complexity, we note that the Algorithm 7, used to fuzzify each selector A_i, exploits at most $|T_r|$ instances. If we consider that the size of the rule is bounded by the number of attributes M, our algorithm complexity is $O(M|T_r|)$ whereas that of [97] is $O(M^2|T_r|)$.

4.3.2 Possibilistic Rule-based Reasoning: PRR

Given a test instance a_{ts} to classify with the following attributes: $(a_1, ..., a_M)$, possibilistic rule based classification [26] consists in estimating the relevance possibility of each rule to the test instance and in assigning the rule output with the highest possibility for the instance a_{ts} calculated as follow:

$$\Pi(r_j | a_{ts}) = \frac{\Pi(a_{ts} | r_j) * \Pi(r_j)}{\Pi(a_{ts})} \tag{4.5}$$

Since $\Pi(a_{ts}) = 1$, only the numerator is useful for comparing rule relevance possibilities. It measures the potential relevance of a rule given the instance to classify. We assume conditional independence for the fuzzy selectors inside the rules. The possibility distribution in the numerator can easily be built by the *product* or the *minimum* of conditional possibilities $\Pi(a_i | r_j)$.

$$\Pi(r_j|a_{ts}) = \prod_{i=1}^{M} \Pi(a_i|r_j) * \Pi(r_j) \tag{4.6}$$

The possibility of attribute a_i given the rule r_j can be estimated by the membership function $\mu_{A_i^F}$ e.g. $\Pi(a_i|r_j) = \mu_{A_i^F}(a_i)$, where A_i^F is the selector i in r_j with the same attribute name as a_i. Prior possibility of each rule $\Pi(r_j)$ is estimated through the rule *certainty factor* CF introduced here as a weight for each rule. Ishibuchi and Yamamoto [101] showed that weighted rules allows to improve classification accuracy of rules. We note that adding weights to rules helps to distinguish between satisfied rules in the multiple classification case.

$$CF(r_j) = \frac{\sum_{a \in T_r^+} \Pi(r_j|a)}{\sum_{a \in T_r} \Pi(r_j|a)} \tag{4.7}$$

where $\Pi(r_j|a)$ is computed as in Equation 4.6 without considering prior possibility for r_j.

The rule chosen by the classifier to classify the current instance, is the one having the highest possibility:

$$c^* = arg \max_{r_j} \Pi(r_j|a_{ts}) \tag{4.8}$$

In this work, we propose three possibilistic rule based classifiers each of them corresponds to each fuzzification approach (A, B or C). In the following we note respectively these classifiers PRC_A, PRC_B, and PRC_C [26].

Example 4.1.

This example illustrates the process of rule evaluation by the Possibilistic Rule based Reasoning. For this reason, we investigate to test it on a fuzzy rule set extracted by the fuzzification module (shown in Figure 4.1) for the "Iris" dataset [23]. We recall that this data set contains four continuous attributes and three categorical classes. For simplicity, we note respectively attributes: SL, SW, PL and PW and classes: Set, Col and Gin. Characteristics for this dataset are given in Table 5.1 and its normalized version in Table 5.2.

Let consider the following crisp rule set generated by the standard learning process of the PART algorithm.

$r_1 : IF(PW \leq 0.208)\ Then\ "Iris - setosa"$

$r_2 : IF(PL \leq 0.627)\ Then\ "Iris - versicolor"$

$r_3 : IF(PW > 0.667)\ Then\ "Iris - virginica"$

$r_4 : IF(PL > 0.661), (PW \leq 0.583)\ Then\ "Iris - virginica"$

We assume that, if confronted to a training set, these rules have the following certainty factors (CF) computed using Formula 4.7 and given in Table 4.1. We also give the fuzzy version for each rule selector obtained by applying the fuzzification algorithm. For simplicity, we only consider the fuzzification *ApproachC* in this example. Thus, only the *support* for each rule selector is given in Table 4.1, the core is equal to the crisp value of each selector.

Table 4.1: Fuzzy rules and their CF

Rule	CF	Selector	*Core*	*Support*
r_1	1	$(PW \leq 0.208)$	0.208	0.375
r_2	0.463	$(PL \leq 0.627)$	0.627	0.644
r_3	0.958	$(PW > 0.667)$	0.667	0.625
r_4	0.816	$(PL > 0.661)$	0.661	0.644
		$(PW \leq 0.583)$	0.583	0.917

Let consider the three following test instances presented in Table 4.2:

Table 4.2: Test instances to classify

Instance:	**SL**	**SW**	**PL**	**PW**	**Class**
a	0.778	0.417	0.831	0.833	?
a'	0.167	0.208	0.593	0.665	?
a''	0.083	0.5	0.068	0.042	?

Following equation 4.6, we obtain:

$\Pi(r_1|a) = \Pi(PW = 0.833|r_1) * \Pi(r_1) = 0 * 1 = 0$ (0.833 is outside the support of r_1 selector)

$\Pi(r_2|a) = \Pi(PL = 0.831|r_1) * \Pi(r_2) = 0 * 0.463 = 0$ (0.831 is also outside the support of r_2 selector)

$\Pi(r_3|a) = \Pi(PW = 0.833|r_3) * \Pi(r_3) = 1 * 0.958 = 0.958$ (r_3 is strongly satisfied by a since 0.833 in side the core of r_3 selector)

$\Pi(r_4|a) = \Pi(PL = 0.831|r_4) * \Pi(PW = 0.833|r_4) * \Pi(r_4) = 1 * 0.25 * 0.816 = 0.205$

If we use the crisp version of r_4, we can see that this rule do not cover this example (especially the second selector) since the value PW=0.833 is outside the boundaries of the second rule selector. However, rule fuzzification enables to r_4 to be relevant to

some degree with a possibility > 0 since $\Pi(PW = 0.833|r_4)) = \mu^F_{(PW \leq 0.583)}(0.833) = \frac{0.917-0.833}{0.917-0.583} = 0.25$ ($\mu^F_{(PW \leq 0.583)}$ is the membership function for $(PW \leq 0.583)$).

Now let consider the second instance a':

$\Pi(r_1|a') = \Pi(PW = 0.665|r_1) * \Pi(r_1) = 0 * 1 = 0$

$\Pi(r_2|a') = \Pi(PL = 0.593|r_2) * \Pi(r_2) = 1 * 0.463 = 0.463$

$\Pi(r_3|a') = \Pi(PW = 0.665|r_3) * \Pi(r_3) = 0.952 * 0.958 = 0.912$

(Since $\Pi(PW = 0.665|r_3) = \frac{0.665-0.625}{0.667-0.625} = 0.952$ by Formula 4.1 applied to the left part of the membership)

$\Pi(r_4|a') = \Pi(PL = 0.593|r_4) * \Pi(PW = 0.665|r_4) * \Pi(r_4) = 0 * 0.754 * 0.816 = 0$

(Since $\Pi(PW = 0.665|r_4) = \frac{0.917-0.665}{0.917-0.583} = 0.754$ by Formula 4.1 applied to the right part of the membership)

Following the same process, we computed the possibility measures for the third instance a''. Rule relevance possibilities given test instances are summarized in Table 4.3 (Column PRC_C for each rule). For the PART algorithm, rules are evaluated in a binary manner using $true/false$ values desponding on if the crisp rule covers the current instance or not.

Table 4.3: Results of the rule evaluation by PART and the PRC

Rules	r_1		r_2		r_3		r_4		Class	
	Part	PRC_C	Part	PRC_C	Part	PRC_C	Part	PRC_C	Part	PRC_C
a	false	0	false	0	true	0.958	false	0.205	Gin	Gin
a'	false	0	true	0.463	false	0.912	false	0	Col	Gin
a''	true	1	true	0.463	false	0	false	0	Set	Set

By analyzing results in Table 4.3, we note that using the original crisp rule set, only one rule covers the first instance a. Rule fuzzification, used by the PRC, causes a multiple classification of a mono-classified by PART (r_3 and r_4 has a possibility degree > 0). In spite of the multiple-covering case, the PRC shows a high ability to distinguish between the strongly satisfied rule (r_3) and that which is weekly satisfied (r_4). The classification for the PART and the PRC coincides.

When classifying the third instance a'', at the origin we are faced to a multiple classification case where more than one rule covers the instance (r_1 and r_2). Applying the decision list principal, PART will use the first satisfied rule (r_1) to classify this instance. We can see that the PRC also succeeds to distinguish between the two satisfied rules using the PRR principal. The two classifiers also coincides in this case

and assign this instance to the class $'Setosa'$.

However, for the second instance a' the two classifiers produce different decisions. Since the PART considers only satisfied rules, it will classify a' as $'Iris-versicolor'$ (the output of r_2) without looking for remaining rules. However, the PRC estimates the relevance possibility of rules in a more flexible way. Although r_3, in its crisp version, do not cover instance a', it has a high possibility to be relevant for this instance since the attribute value (PW = 0.665) is too near to the core ($C_L = 0.667$) of the rule selector that's why it has a high degree of membership. Moreover, this rule has also a high certainty factor. So if compared to r_2, r_3 seems to be more relevant with a high possibility degree than that of r_2 even it is "slightly" not satisfied. The two classifiers diverge for this instance, this divergence in classification enables to the PRC to be more efficient when classifying instances with attribute values near to the decision boundaries of rules as will be shown in the experimental section.

4.4 The Hybrid Possibilistic Rule-based Classifiers: HPRCs

In the multiple classification case, the main problem of unordered rule selection is that classification accuracy may significantly deteriorate if the classifier fails to distinguish between satisfied rules. Mainly for the PRC_B and in some particular cases, rules may have too close plausibility estimates even if the PRC include weights to be able to distinguish between satisfied rules as in Equation 4.6. To deal with this problem, we investigate the idea of integrating the PRC_B with the PRC_A in a hybrid classifier. In this later case, if the main classifier based on the approach B fails to distinguish between the most relevant rules, we expect that the classifier based on approach A will help to distinguish between these rules and thus better separate the conflicting situations. Furthermore to deal with the non-covering examples for the PRC_B, we also investigate the hybridizing with the PRC_C. The fuzzification approach C, by using large support, is able to cover examples not covered by any rule in approach B. The Hybrid classifier denoted $PRC_{(B+A+C)}$ is based on the following algorithm [26]:

If r_1 and r_2 are respectively the most and the second most relevant rules for an instance a_{ts}, the classification ambiguity with respect to rules is defined by:

$$Ambiguity(a_{ts}, r_1, ..., r_n) = 1 - (\Pi(r_1|a_{ts}) - \Pi(r_2|a_{ts})) \tag{4.9}$$

In this algorithm we can also estimate ambiguity by considering more than two rules. In practice, au maximum three levels is sufficient for disambiguation even for data sets with high number of classes.

Algorithm 8 The Hybrid Possibilistic Rule based Classifier $PRC_{(B+A+C)}$ (a_{ts})

Classify a_{ts} by PRC_B
$c_1 \leftarrow$ Class of the most plausible rule r_1 in the PRC_B (Given by Formula 4.8)
$c_2 \leftarrow$ Class of the second most plausible rule r_2
if $(\neg MultipleClassification$ OR $Ambiguity(a_{ts}, r_1, r_2) < \varepsilon)$ **then**
  Class$(a_{ts}) = c_1$
else
  Classify a_{ts} by PRC_A
  Class(a_{ts})= Class of the most plausible rule r_1 in the PRC_A
end if
if $non - covering - case$ **then**
  Classify a_{ts} by PRC_c
  Class(a_{ts})= Class of the most plausible rule r_1 in the PRC_C
end if
return Class(a_{ts})

4.5 Experiments and discussion of Possibilistic Rule-based Classifiers

In order to test the performance of the proposed possibilistic approaches, we implemented and tested the Possibilistic Rule-based Classifier (with the three versions) under the JAVA language. For testing the PART algorithm, we used the WEKA implementation of this algorithm [168] (within java language).

In addition to the well known advantages of oriented object technology, this last has several qualities relative to our needs. Indeed, the integration of software components in the applications developed in this environment is usually a simple action. In one hand, the JBuilder environment has allowed to us to integrate a fundamental WEKA component for our application (See Appendix A). In particular, the class for data management which allows using and analyzing all data sets having an "arff" type and also the class PART() corresponding to the PART algorithm available in the Weka software [168]. Once integrated, we could properly structure and adapt theses classes in order to be extended to the fuzzy version of the PART algorithm to response to the needs of our classifier. In the other hand, this environment allows the development of a JAR package which can be easily integrated in any single-user or other Web applications.

This section is divided into parts, we first present the characteristics of data sets used to conduct experiments. Then we give experimental results for Possibilistic Rule-based Classifiers as a fuzzy extensions for the PART algorithm.

4.5.1 Data sets

The experimental study is based on several data sets taken from the U.C.I repository of machine learning databases [131]. A brief description of these data sets is given in Table 4.4. Since we have chosen to deal only with numerical attributes in this study, all these data sets have numerical attribute values.

Table 4.4: Description of datasets

Database	**Data**	**Attributes**	**Classes**
Iris	150	4	3
W. B. Cancer	699	8	2
Wine	178	13	3
Diabetes	768	7	2
Magic gamma telescope	1074	10	2
Transfusion	748	4	2
Satellite Image	1090	37	6
Segment	1500	20	7
Yeast	1484	9	10
Ecoli	336	8	8
Glass	214	10	7
Iosophere	351	35	2
Letter	3050	17	26
Block	5473	10	5
German	1000	25	2
Heart	270	14	2

For each data set, we used a 10-cross-validation to compare the accuracy of the classifiers.

4.5.2 Classification results

This section provides experimental results of Possiblistic Rule based Classifiers [26].

Table 4.5 shows the classification performance obtained by the four proposed classifiers for the 14 mentioned data sets. We also included the classification accuracy of the PART as well as the FURIA algorithm (as presented in [97]) for a comparative purpose. The number of induced rules is also given for the PRC_s and FURIA [97]. In this experimental study, we have applied an aggregation based on the *minimum* for the approach A and C and the *product* for approach B. In order to fix the best ambiguity level used for the hybrid approach, we have tested different values for this threshold (0, 0.1, ..., 0.9, 1) for each dataset and then choose the optimal value (which maximizes accuracy). The best ambiguity level is 0.1 for almost all datasets.

Table 4.5: Classification accuracies given as the mean and the standard deviation of 10 cross-validations and average number of rules

	PRC_A	PRC_B	PRC_C	PRC_{B+A+C}	$PART$	$FURIA$	$Rules_{(PRC_s)}$	$Rules_{(FURIA)}$
Iris	**96.67±4.47**	96.0±4.42	**96.67±4.47**	96.0±4.42	94.67±4.99	94.76	3.1	4.4
Cancer	95.46±3.31	95.91±2.84	95.17±2.54	**96.64±2.46**	94.88±3.1	95.68	8.6	12.2
Wine	**94.93±3.9**	93.26±4.14	90.48±6.56	93.26±4.14	93.82±4.61	93.25	3.3	6.2
Diabetes	74.22±3.79	74.62±4.74	**75.53±5.46**	75.14±5.57	74.22±4.31	74.71	6.6	8.5
Magic	77.19±4.47	78.4±3.36	**79.24±3.21**	78.68±3.56	78.96±2.53	-	8.9	-
Transf.	77.15±5.4	77.95±5.23	**78.09±5.33**	**78.09±5.33**	77.69±5.71	-	3.4	-
Sat.Image	91.47±2.87	90.92±4.15	93.12±4.01	91.74±2.99	**93.76±2.12**	-	14.4	-
Segment	89.14±3.32	89.4±3.87	91.93±2.91	89.4±3.87	95.2±1.63	**96.50**	21.9	26.9
Yeast	53.64±3.04	56.54±2.57	**60.51±3.77**	56.54±2.57	54.5±4.3	-	127	-
Ecoli	80.1±5.11	78.63±5.96	73.18±6.53	79.8±6.0	81.53±4.14	**83.12**	12.3	13.8
Glass	69.18±7.26	**70.13±8.1**	62.64±8.3	**70.13±8.1**	69.68±7.97	68.22	13.6	11.3
Iosophere	88.03±4.92	89.75±3.87	**90.31±3.44**	**90.31±3.44**	89.45±4.45	89.59	6.8	8.3
German	74.4±3.44	74.0±3.58	74.0±3.32	**74.6±3.5**	71.6±3.04	-	73.4	-
Heart	79.26±8.32	79.63±7.64	**81.48±5.74**	79.63±8.49	77.78± 8.28	79.75	15.6	8.4

This means that the hybrid classifier outperforms the classifier B only if the conflict between rules is very high.

By comparing the classification performance in Table 4.5 we note that:

• The PRC_C is more accurate than PRC_A and PRC_B in nine of 14 data sets and less accurate in the remaining data sets. However, PRC_B seems to have the second rank since it is more accurate than PRC_A in 9 of 14 data sets.

• Results for the hybrid version shows that either it contributes to improve the accuracy of the original classifier PRC_B or at least keeps its performance. Indeed, for 8 of 14 data sets the $PRC_{(B+A+C)}$ increases the accuracy of the PRC_B and for 6 data sets the two classifiers have the same accuracy. This shows the efficiency of the hybrid approach to separate conflicting rules and helps the PRC_B to choose the correct class.

• When the hybrid classifier is equivalent to the PRC_B, this means that the approach B and A always predict the same class. Then, for this data set there is no non-covering examples (the learned rule set covers all testing examples), or the approach C predicts the same class as the default class of approach B for non-covering examples.

• Overall, the proposed possibilistic rule based classifiers (PRC_A or PRC_B or PRC_C or $PRC_{(B+A+C)}$) significantly outperforms the PART classifier in 11 of 14 data sets,

the highest increase is for the “Yeast”, “German” and “Heart”.

• If we compare the accuracy of the proposed classifiers to the FURIA, we can see that the PRC outperforms FURIA in 7 of 9 tested data sets and is less accurate in the two remaining data sets ”Segment” and ”Ecoli”. For these data sets (and the Sat.Image), we also remark that our proposed classifiers are also less accurate than PART algorithm. In fact, for data sets with a large number of classes, there is a high risk of multiple classification mainly if the number of generated rules by PART is also large (see rules for these datasets in Table 4.5). Besides, if training instances are not equitably distributed over classes, this will cause the generation of some rules that are very robust for classes with high coverage and other non robust rules for classes with very low coverage. Considering that the proposed classifiers use unordered rules, they will always favor robust rules and neglect others in the multiple classification case, which causes classification error.

• Comparison in terms of the number of rules in Table 4.5 shows that the PRC_s use a reduced rule set with an average over all datasets equal to 10.2 against to 11.11 for the FURIA.

Finally, in order to measure the improvement significance of the PRC if compared to the PART algorithm in terms of accuracy, we used the Wilcoxon Matched-Pairs Signed-Ranks Test as proposed by Demsar [50]. It is a non-parametric alternative to the paired t-test that enables us to compare two classifiers over multiple data sets. Comparison results give a $p - value <= 0,03271$ which show that the $PRC_{(AorBorC)}$ is significantly better than the PART for all data sets.

These results prove that using unordered fuzzy rules is faithful and contributes to improve the efficiency of a classical decision list [26].

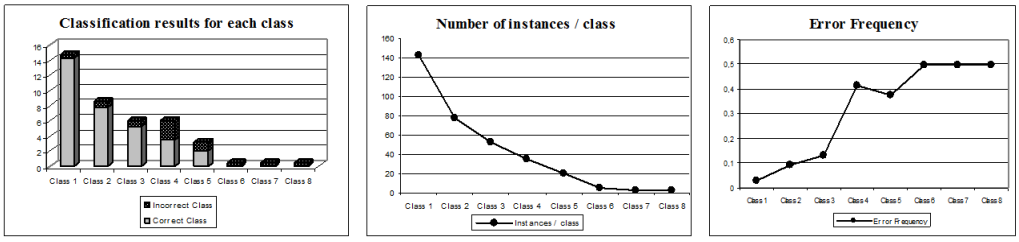

Figure 4.4: Error frequency by class for the Ecoli data set

4.6 Conclusion

In this chapter we have proposed and validated the performance of a family of fuzzy rule-based classifiers called PRCs which is an extension and a modification of the PART algorithm. The PRCs differ from the PART algorithm mainly in three ways.

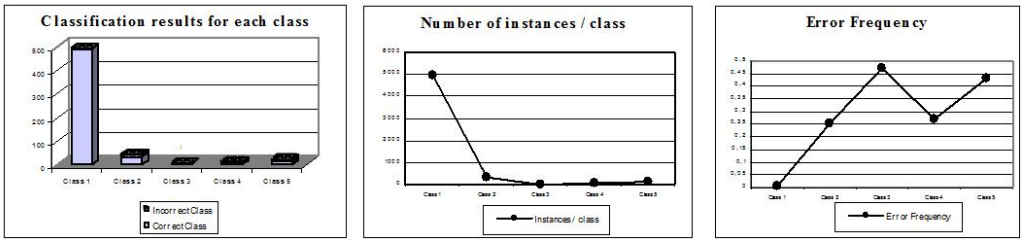

Figure 4.5: Error frequency by class for the Block data set

In particular, this classifier uses fuzzy rules instead of crisp rules to give more flexibility to rule decision boundaries. Moreover, it considers rules in an unordered manner instead of decision lists at classification step. Finally, using fuzzy rules with large supports enables us to cover all non covered examples without using a prefixed default class.

We also propose a Hybrid Possibilistic Rule based Classifier which integrates different fuzzification approaches to deal deal with different situations in the Possibilistic Rule-based Reasoning.

Experimental results show the interest of Possibilistic Rule-based Classifiers. In fact, the modifications added to the PART algorithm, mainly the use of unordered and fuzzy rules instead of crisp and rule lists enables to the PRCs to significantly outperforms the classical PART algorithm.

Chapter 5

Possibilistic Classifiers for perfect /imperfect numerical data

5.1 Introduction

Classification is a machine learning technique used to predict group membership for data instances. It consists in searching for algorithms that produce general classifiers from a set of training instances which constitutes the training phase. The resulting classifier is then used to assign class labels to the testing instances described by a set of predictor features. This process is usually called testing phase. Classification tasks can be handled by mainly three classes of approaches: those based on empirical risk minimization (decision trees [146], artificial neural networks [13]), approaches based on maximum likelihood estimation (such as Bayesian networks [139], k-nearest neighbors [43]), and the ones based on Kolmogorov complexity [160].

In this dissertation we are mainly interested in the second class of methods. Given a new piece of data to classify, this family of approaches seeks to estimate the plausibility of each class with respect to its description (built from the training set of examples), and assigns the class having the highest plausibility value. There are principally two methods: the k-Nearest Neighbors and the Naive Bayesian Classifiers (NBC). The former, known as lazy learning methods, are based on the principle that an instance will usually exist in the proximity of other instances having similar characteristics. The latter (NBC type) assumes independence of variables (attributes) in the context of classes in order to estimate the probability distribution on the classes for a given observed data. NBCs are also known for their simplicity, efficiency and small needs in terms of storage space. Moreover NBC perform well, even when making the strong independence assumption which is almost always violated in real datasets [54].

The objective of this work is to discuss the benefits and also the limits of Naive

Bayesian Classifier and to test the feasibility of using other kinds of representation for the distributions associated to attributes. This work focuses on the classification of data with *numerical* attributes. Three alternatives are commonly considered for handling numerical attributes in an NBC: i) using a discretization process for continuous attributes and then applying a multinomial probability distribution. It may lead to a loss of information [172] mainly when attributes are discretized in many intervals. However, this method may be effective when the elicitation of the density function turns to be difficult; ii) assuming normality of the distributions for attributes and estimating the density function using Gaussian densities, or iii) directly estimating densities in a non parametric way using kernel density functions.

The study of possibilistic classifiers is motivated by the good performance of NBCs and their appropriateness for incomplete data in the first hand and by the ability of possibility theory [70] to handle poor and imperfect data in the second hand.

Imperfection in databases, including imprecision and uncertainty, is gaining more and more attention since decision systems are not able to deal with such kind of information. Thus, it is required for a classification technique to be able to model and deal with instance imperfections in order the induced model can be a real representation of data. Several non-classical theories of uncertainty have been proposed in order to deal with uncertain and imprecise data such as, evidence theory [156], fuzzy set theory [173] and possibility theory [70] [63]. The latest is able to deal with different kinds of uncertainty and is useful for representing partial knowledge or incompleteness.

Possibility theory [70] [63] has been recently proposed as a counterpart of probability theory to deal with classification tasks in presence of uncertainty. There are only few works that treat possibilistic classification [10] and most of existing naïve possibilistic classifiers deal only with categorical attributes.

In this chapter, we will first introduce the basics for the Naive Possibilistic Classifiers (NPC) that are based on the possibilistic counterpart of the Bayesian formula [71] and also propose the estimation of the possibility distributions for *numerical* data.

In the second time, we also investigate the idea of integrating the possibilistic classifiers with a Nearest Neighbors based Heuristic (NNH) in a hybrid manner in order to improve their performances. Indeed, the hybrid classification allows the use of the NNH as an alternative when the main classifier fails to distinguish between classes, i.e. when several classes have very close plausibility estimates.

In the last part of this chapter, we study possibilistic classifiers applied to $imperfect$

numerical data. The objective of this part is to extend possibilistic classifiers previously proposed for numerical data in order to cope with uncertainty in data sets (in the training and testing sets).

The rest of the chapter is structured as follows: in the next section, we provide a summary of related works on classification under imperfect data. In Section 3, we give our motivation to the possibilistic classification task and we restate the basic setting of this classification method. In Section 4, we study the two kinds of elicitation methods for building possibility distributions: i) the first one is based on a transformation method from probability to possibility, whereas ii) the second one makes a direct, fuzzy histogram-based, or possibilistic interpretation of data, taking advantage of the idea of proximity. Section 5 introduces the principle of the hybrid classification. In Section 6, we extend possibilistic classifiers to handle imperfect data: the first type concerns the processing of *uncertain* classes in the training set, whereas the second one deals with *uncertain* attribute values in the testing set. Uncertainty on attribute values is modeled by intervals.

5.2 Related Works

Some approaches have already proposed the use of a possibilistic data representation in classification methods based on decision trees, Bayesian-like, or case-based approaches. A general discussion about the appropriateness of fuzzy set methods in machine learning can be found in [99]. Most of the works in possibilistic classification are motivated by the handling of imprecision and uncertainty about attribute values or the classes. Some assume that there is a partial ignorance about class values. This ignorance, modeled through possibility distributions, reflects the expert knowledge about the possible class of the training instances.

In general, the approaches deal with discrete attribute values only and are not appropriate for continuous attributes (and thus require a preliminary discretization phase for handling continuous attribute values).

By contrast, the work reported here presents a new type of classification method suitable for training data pervaded with uncertainty. It relies on a possibilistic representation of the attribute values associated to a class, which offers more flexibility than the classical probabilistic setting. Moreover, we focus on the handling of numerical attributes. Such possibilistic model is then extended to support uncertainty related to the class and attributes.

We now provide a brief survey of the literature on possibilistic classification. We start with approaches based on decision trees, before a more detailed discussion on Bayesian classifiers applied to possibilistic data.

Denoeux and Zouhal [51] use possibility theory to model and deal with uncertain labels in the training set. To do this, the authors assign a possibility degree to each possible class label which reflects the possibility that the given instance belongs to this class. Besides, Ben Amor et al. [3] have developed a qualitative approach based on decision trees for classifying examples having uncertain attribute values. Uncertainty on attribute values is represented by means of possibility distributions given by an expert. In [103], possibilistic decision trees are induced from instances associated with categorical attributes and vaguely specified classes. Uncertainty, modeled through possibility theory, concerns only the class attribute whereas other predictive attributes are supposed to be certainly known. The authors developed three approaches for possibilistic decision trees. The first one, using possibility distributions at each step of the tree construction, is based on a measure of non-specificity in possibility theory in order to define an attribute selection measure. The two remaining approaches make use of the notion of similarity between possibility distributions for extending the C4.5 algorithm in order to support data uncertainty.

A naive Bayesian-like possibilistic classifier has been proposed by Borgelt et al. [16] to deal with imprecise training sets. For this classifier, imprecision concerns only attribute values of instances (the class attribute and the testing set are supposed to be perfect). Given the class attribute, possibility distributions for attributes are estimated from the computation of the maximum-based projection [18] over the set of precise instances which contains both the target value of the considered attribute and the class.

A naive possibilistic network classifier proposed by Haouari et al. [93], presents a procedure that deals with training datasets with imperfect attributes and classes, and a procedure for classifying unseen examples which may have imperfect attribute values. This imperfection is modeled through a possibility distribution given by an expert who expresses its partial ignorance, due to a lack of a priori knowledge. There are some similarities between our proposed approach and the one by [93]. In particular, they are both based on the idea stating that an attribute value is all the more possible if there is an example, in the training set, with the same attribute value in the discrete case, or a very close attribute value in terms of similarity in the numerical case. However, the approach in [93] does not require any conditional distribution over attributes to be defined in the certain case, whereas a preliminary requirement in our approach, is to estimate such a possibility distribution for numerical data in the certain case.

Benferhat and Tabia [10] propose an efficient algorithm for revising, using Jeffrey's rule, possibilistic knowledge encoded by a naive product-based possibilistic network classifier on the basis of uncertain inputs. The main advantage of the proposed algorithm is its capability to process the classification task in polynomial time with respected to the number of attributes.

In [143], the authors propose a new Bayesian classifier for uncertain categorical or continuous data by integrating uncertainty in the Bayesian theorem and propose a new parameter estimation method. An attempt to treat uncertainty in continuous data is proposed in [144], where authors developed a classification algorithm able to generate rules from uncertain continuous data. For the two works [144], [143], uncertainty over continuous attribute values is represented by means of intervals. This imprecision is handled by a regular probabilistic process.

Besides, some case-based classification techniques, which make use of possibility theory and fuzzy sets, are also proposed in the literature. We can particularly mention the possibilistic instance-based learning approach [98]. In this work, the author proposes a possibilistic version of the classical instance-based learning paradigm using similarity measures. Interestingly, this approach supports classification and function approximation at the same time. Indeed, the method is based on a general possibilistic extrapolation principle that amounts to state the more similar to a known example the case to be classified is, the more plausible the case and the example should belong to the same class. This idea is further refined in [98] by evaluating this plausibility by means of an interval whose lower bound reflects the "guaranteed" possibility of the class, and the upper bound the extent to which this class is not impossible. In a more recent work [12], the authors develop a bipolar possibilistic method for case-based learning and prediction.

This possibilistic instance-based learning approach may look similar to the proximity-based classifiers proposed in [23]. However, there are differences, although both emphasize a possibilistic view of classification based on similarity. In [98] a conditional possibility of a class given the case description is defined directly, taking into account all the attributes together. In the methods presented in [23], we rather start by defining the plausibility of a particular attribute value for a given class (on a similarity basis), and then apply a Bayesian-like machinery for obtaining the classification result. The fact that the similarity idea is applied to attributes one by one, makes then necessary to resort to an independence assumption.

5.3 General setting of possibilistic classification

The idea of applying possibility theory to classification parallels the use of probabilities in Bayesian classifiers (see Chapter 2 for a reminder). Probability distributions used in NBCs are usually built by assuming that numerical attributes are normally distributed around their mean. Even if a normal distribution is appropriate, identifying it exactly from a sample of data is especially questionable when data are poor. When normality assumption is violated, Gaussian kernels can be used for approximating any type of distributions. Then, it is required to assess many parameters, a

task that may be not compatible with poor data. The problem of the precise estimation of probability distributions for NBCs is important for the exact computation of the probability distribution over the classes. However, due to the use of the product for combining probability values (which are often small), the errors on probability estimations may have a significant effect on the final estimation. This contrasts with possibility distributions which are less sensitive to imprecise estimation for several reasons. Indeed, a possibility distribution may be viewed as representing a family of probability distributions corresponding to imprecise probabilities, which sound more reasonable in case of poor data. Moreover, we no longer need to assume a particular shape of probability distribution in this possibilistic approximation process.

As in the case of Bayesian classification, possibilistic classification is based on the possibilistic version of the Bayes theorem. Given a new vector $a = \{a_1, ..., a_M\}$ of M observed variables $A_1, ..., A_M$ and the set of classes $C = \{c_1, ..., c_C\}$, the classification problem consists in estimating a possibility distribution on classes and in choosing the class with the highest possibility for the vector a in this quantitative setting, i.e.:

$$\Pi(c_j|a_1, ..., a_M) = \frac{\pi(c_j) * \pi(a_1, ..., a_M|c_j)}{\pi(a_1, ..., a_M)} \tag{5.1}$$

In formula (5.1), the quantitative component of possibilistic classification involves *prior* possibility distribution relative to the class and the input vector. Note that the term $\pi(a_1, ..., a_M)$ is a normalization factor and it is the same over all class values. In this work, we assume that there is no a priori knowledge about the input vector to classify (thus $\pi(a_1, ..., a_M) = 1$)). Moreover, as in naive Bayesian classification, naive possibilistic classification assumes that variables A_i are independent in the context of classes [5].

Assuming attribute independence, the plausibility of each class for a given instance is computed as:

$$\pi(c_j|a_1, ..., a_M) = \pi(c_j) * \prod_{i=1}^{M} \pi(a_i|c_j) \tag{5.2}$$

where conditional possibilities $\pi(a_i|c_j)$ in formula (5.2) represent to what extent a_i is a possible value for the attribute A_i in the presence of the class c_j. As in the case of the conditioning rule, $*$ may be chosen as the *min* or the *product* operator (min corresponds to complete logical independence, while the use of the product makes partially possible values jointly less possible). In a product-based setting, a given instance is assigned to the most plausible class c^*:

$$c^* = arg \max_{c_j} \Pi(c_j) * \prod_{i=1}^{M} \Pi(a_i|c_j) \tag{5.3}$$

Possibilistic classification is based on the assumption that there always exist an observed attribute value in the training set which forces $\Pi(a_i|c_j) = 1$. This assumption is called the *postulate of observation relevance* [71].

Moreover, it is worth noticing that formula (5.2) has a set-theoretical reading. Namely, when the possibility distributions take only the values 0 and 1, the formula (5.2) amounts to express that an instance may be possibly classified in c_j inasmuch as the attribute value of this instance are compatible with this class given the available observations. Thus, possibilistic classification may be viewed as an intermediate between Bayesian probabilistic classification and a purely set-based classifier (such classifiers use as distributions the convex hull for each attribute of the data values to identify classes, usually leading to too many multiple classifications).

5.4 Possibilistic distributions for perfect numerical data

In this section, we describe several methods for building possibility distributions from data belonging to continuous domains. In the first part of this section, we consider two families of approaches: the first one is based on a probability-possibility transformation method [74], [58], [171]. The second one is based on a direct possibilistic interpretation of data taking advantage of the idea of proximity. In the second part, we study the ability of these possibilistic classifiers to detect ambiguity between near classes and we propose a hybrid approach for this purpose. All these reflections give us the intuition for other variants of approaches for building possibility distributions. Following this line, we propose to compute possibility distributions as a family of Gaussian probabilistic distributions.

For all the rest of this work, all attribute values a_i's are normalized as follows:

$$a_{i_n} = \frac{a_i - min(a_i)}{max(a_i) - min(a_i)}, \tag{5.4}$$

min and max are functions giving respectively the minimum and the maximum value of the attribute a_i over the training set. For the sake of simplicity we use in the rest of the chapter only normalized attribute values, e.g., every attribute value a_i refers to the corresponding a_{i_n}.

5.4.1 Probability to possibility transformation based classifiers

The transformation from probability to possibility distributions [58], which has been extended to continuous universes, accounts for epistemic uncertainty. It yields the most restrictive possibility distribution that is co-monotone with the probability distribution and that provides an upper-bound on the probability of any event.

In this section, we propose two elicitation approaches [23] [24] based on the probability to possibility transformation method, previously described in Section 1.5.1 page 36. We apply this transformation method to a Gaussian distribution, which leads to two classifiers called $Naive\ Possibilistic\ Classifier$ and the $Flexible\ Naive\ Possibilistic\ Classifier$.

Gaussian density-based transformation: the Naive Possibilistic Classifier (NPC)

Let consider the training set Tr composed of N instances involving M numerical attributes. We have to find a possibility distribution over a training set which is the most specific representation for the numerical data. If we apply equation (1.24) page 38 to our classification problem, $\pi(a_i|c_j)$ can be estimated by: $1-P(I_{a_i}|c_j)$ where I_{a_i} is the confidence interval as previously presented. The main question is to estimate such confidence interval for each attribute a_i belonging to the class c_j.

For the NPC, we assume that each attribute value a_i is a random variable which is normally distributed over the class c_j. Thus for each class c_j, a Gaussian distribution $g_{ij} = g(a_i, \mu_{ij}, \sigma_{ij})$ should be given. For such Gaussian, μ_{ij} is the mean of the variable a_i for the class c_j and σ_{ij} is its standard deviation for the same class.

If I_{a_i} is the confidence interval centred at μ_{ij}, its probability $P(I_{a_i}|c_j)$ can be estimated by:

$$P(I_{a_i}|c_j) = 2 * G(a_i, \mu_{ij}, \sigma_{ij}) - 1. \tag{5.5}$$

where G is a Gaussian cumulative distribution easily evaluated using the table of the Standard Normal Distribution. We propose to estimate $\pi(a_i|c_j)$ by $1-P(I_{a_i}|c_j)$ using the following formula:

$$\pi(a_i|c_j) = 1 - (2 * G(a_i, \mu_{ij}, \sigma_{ij}) - 1) = 2 * (1 - G(a_i, \mu_{ij}, \sigma_{ij})). \tag{5.6}$$

Hence, in the training phase we should simply calculate the mean μ_{ij} and the standard deviation σ_{ij} for each attribute a_i of instances belonging the class c_j.

Example 5.1.

Let us consider the subset T_r of training data taken from the "*iris*" data set [23] in which 5 examples are given for each iris category as shown in Table 5.1.

Table 5.1: Original training set

Instance K	**SL**	**SW**	**PL**	**PW**	**Class**
1	5.10	3.50	1.40	0.20	**Set**
2	4.90	3.00	1.40	0.30	**Set**
3	4.70	3.20	1.30	0.20	**Set**
4	4.60	3.10	1.50	0.50	**Set**
5	5.00	3.60	1.40	0.10	**Set**
6	5.60	3.20	4.70	1.40	**Col**
7	6.40	3.20	4.50	1.50	**Col**
8	6.90	3.10	4.90	1.50	**Col**
9	5.50	2.30	4.00	1.30	**Col**
10	6.50	2.80	4.60	1.50	**Col**
11	6.30	3.30	6.00	2.50	**Gin**
12	5.80	2.70	5.10	1.90	**Gin**
13	7.10	3.00	5.90	2.10	**Gin**
14	6.30	2.90	5.60	1.80	**Gin**
15	6.90	3.00	5.80	2.20	**Gin**
Min	4.6	2.30	1.30	0.20	
Max	7.1	3.60	6.00	2.50	

This example illustrates how to estimate a possibility distribution for classes given a new instance to classify in the case of NPC. The calculus process of probability measures, used in this chapter as a dual measure to possibilities, is also explained. Attributes in the training set T_r are normalized so that every attribute value is in [0,1] according to equation (5.4). The corresponding normalized training set is given in Table 5.2. In this example (and in Example 5.2, 5.3 and 5.4 in the following sections), we only consider the perfect case where each training instance is perfectly classified in a particular class.

Table 5.3 presents the calculus results of the mean and standard deviation for each attribute and class.

Given a new instance to classify (Table 5.4), we will show a detailed computation of conditional possibilities of only one attribute, namely SL. To estimate the possibility of $(SL = 0.63|Set)$ one should start by identifying the confidence interval for this attribute value. Considering that $0.63 > 0.104$ (0.104 is the mean of SL for the class Set in Table 5.3), the confidence interval is $I(SL = 0.63) = [-0.422, 0.63]$. Let us compute the probability of this confidence interval using equation (5.5):

Table 5.2: Training set with normalized attributes

Instance k	**SLn**	**SWn**	**PLn**	**PWn**	**Class**
1	0.20	0.92	0.02	0.04	**Set**
2	0.12	0.54	0.02	0.08	**Set**
3	0.04	0.69	0.00	0.04	**Set**
4	0.00	0.62	0.04	0.17	**Set**
5	0.16	1.00	0.02	0.00	**Set**
6	0.40	0.69	0.72	0.54	**Col**
7	0.72	0.69	0.68	0.58	**Col**
8	0.92	0.62	0.77	0.58	**Col**
9	0.36	0.00	0.57	0.50	**Col**
10	0.76	0.38	0.70	0.58	**Col**
11	0.68	0.77	1.00	1.00	**Gin**
12	0.48	0.31	0.81	0.75	**Gin**
13	1.00	0.54	0.98	0.83	**Gin**
14	0.68	0.46	0.91	0.71	**Gin**
15	0.92	0.54	0.96	0.88	**Gin**

Table 5.3: The mean and standard deviation of different attributes

		SL	**SW**	**PL**	**PW**
Set	μ	0.104	0.754	0.021	0.067
	σ	0.083	0.20	0.02	0.06
Col	μ	0.632	0.477	0.689	0.558
	σ	0.242	0.295	0.072	0.037
Gin	μ	0.752	0.523	0.932	0.833
	σ	0.209	0.167	0.076	0.114

$P(I_{(SL=0.63)}|Set) = P(-0.422 < SL < 0.63) = \int_{-0.422}^{0.63} g(SL, 0.104, 0.08) d_{SL}$

$P(I_{(SL=0.63)}|Set) = G(\frac{0.63-0.104}{0.08}) - G(\frac{-0.422-0.104}{0.08}) = 2 * G(\frac{0.526}{0.08}) - 1$

From the standard table of Normal distribution, we can evaluate this cumulative Gaussian: $G(6.575) = 1$; *so* $P(I_{(SL=0.63)}|Set) = 1$

Now we can easily derive possibility distribution as a transform of probabilities:

$\pi(SL = 0.63|Set) = 1 - P(I_{(0.63)}|Set) = 1 - 1 = 0$

Similarly we compute the possibility distribution for remaining attributes SW, PL and PW. Table 5.5 summarizes the calculus results of possibility distribution for each attribute and each class.

Table 5.4: Example of instance to classify

	SL	**SW**	**PL**	**PW**
Instance: a	0.63	0.79	0.9	0.81

Table 5.5: Possibility distribution of classes for the NPC

	$\pi(SL = 0.63\|c_j)$	$\pi(SW = 0.79\|c_j)$	$\pi(PL = 0.42\|c_j)$	$\pi(PW = 0.81\|c_j)$	$\pi(c_j\|a)$
Set	**0**	0.855	0	0	0 (0)
Col	0.993	0.288	0.003	1.45 E-11	1.3E-14 (3.9E-13)
Gin	0.558	0.109	0.673	0.837	**0.034(1)**

The last step is to estimate posterior possibilities for each class given the instance a to classify. In this example, we applied the *product* based setting to aggregate conditional possibilities for attributes. An equivalent result can also be obtained using the *min* based setting.

$\Pi(Set|a) = \Pi(SL|Set) * \Pi(SW|Set) * \Pi(PL|Set) * \Pi(PW|Set)$

$= 0 * 0.85591 * 0 * 0 = 0$

$\Pi(Col|a) = 0.993 * 0.288 * 0.003 * 1.45719E - 11 = 1.3E - 14$

$\Pi(Gin|a) = 0.558 * 0.109 * 0.673 * 0.837 = 0.034$

If we normalize these possibility distributions, we obtain:

$c^* = arg\ max_{c_j}(0, 3.9E - 13, \mathbf{1})$

So the instance a will be assigned to the class "$IrisVirginica$".

Kernel density-based transformation: the Flexible Naive Possibilistic Classifier(FNPC)

The FNPC is mainly based on the FNBC. For this classifier, the building procedure is reduced to the calculation of the standard deviation σ. The FNPC is the same as the NPC in all respects, except that it uses a different method for density estimation. Instead of using a single Gaussian to estimate each continuous attribute, we investigate kernel density estimation as in the FNBC.

It is proved in [107] that classifiers based on kernel estimation are more accurate

than Gaussian based estimation to fit non-Gaussian densities. The idea of estimation based on Gaussian kernels (see Section 2.8.2 page 58) may be adapted in the possibilistic context in the spirit of formula (5.5). Haouari et al. in [93] have justified the use of the arithmetic mean function to estimate a possibility distribution for an attribute given the class when dealing with (n individual possibilities over the training set.

If we just consider that we have to combine possibility measures (forgetting how they have been obtained) the natural way to do it is to use a weighted maximum operator [64]. However our problem, as announced in [93], is closer to being an estimation task than a fusion because training instances come from a single random source and not from n independent sources of information. Besides, the authors show that the arithmetic mean function satisfies the three necessary properties allowing it to be an estimation function: *idempotency, commutability and monotonicity* (see [93] for details).

$$\pi(a_i|c_j) = \frac{1}{N_j} \sum_{k=1}^{N_j} \pi(a_i, c_{jk}). \tag{5.7}$$

with:

$$\pi(a_i, c_{jk}) = 2 * (1 - G(a_i, \mu_{ik}, \sigma)). \tag{5.8}$$

where k ranges over the N_j instances of the training set in class c_j and $\mu_{ik} = a_{ik}$.

Various rules are used in the statistical literature for setting the kernel width σ. John and Langley [107] have proved that the Flexible Bayes classifier is strongly consistent if the kernel density estimate satisfies the theorem of strong consistency [53]. In this theorem, two necessary conditions for the width σ of the kernel density estimate must be satisfied: i) $\sigma \to 0$ as $n \to \infty$ and ii) $n\sigma \to \infty$ as $n \to \infty$.

In this section and for all distributions, the standard deviation is estimated by:

$$\sigma = \frac{1}{\sqrt{N}} \tag{5.9}$$

Both σ estimators in formula (5.9) (and Formula 2.8 page 62) satisfy the two conditions of strong consistency theorem. However, we empirically choose to use the estimator in formula (5.9), due to its better performance in experimentations. It may be due to the fact that the density estimated became increasingly local when we consider all training instances (N) instead of considering only those belonging to a specific class (N_j) when estimating σ. The intuition behind this choice is that this

estimator will contribute to have non smoother (rough) kernel densities which may help to reduce overlapping between classes. In fact, for smooth kernels, probabilities for each class could be very close and don't enable a clear distinction between classes which lead to misclassification. We estimate that, if sufficient number of instances is available for each class, small σ (large N) will contribute to increase accuracy. On contrary, if there are few examples for a class, kernels may be too localized to this class.

As will be seen, such a method contributes to improve the classification accuracy on real datasets as it will be seen in the experimental section. This method exploits a statistical view of the neighborhood, since an instance will have a high probability value for a class as soon as its value for each attribute is close to the values of other examples in the class. In the next section, the idea of closeness will be captured by means of a fuzzy set.

Example 5.2.

Let us continue with the previous example in which we consider the same training set and the same instance a to classify. In the following, we illustrate the calculus process of possibility distributions for the $FNPC$. The standard deviation for this classifier is: $\sigma = 1/\sqrt{15} = 0.258$

For this classifier, in order to derive the final possibility distribution for each attribute given the class, one should compute fifteen individual conditional possibilities one for each instance k in the training set. Let us show a detailed computation of conditional possibility of the attribute $SL = 0.63$ of only the first instance ($k = 1$).

As in Example 5.1, in order to estimate $\pi(SL = 0.63|c_{j1} = Set)$, we start by defining the confidence interval for this attribute value. So, we should consider the attribute value $SL = 0.20$ of the first instance (column 1 of line 1 in Table 5.2) as a mean for the first Gaussian kernel, i.e: $\mu_{SL_1} = 0.2$. Since $0.63 > 0.2$, the confidence interval is: $I_{(SL=0.63)} = [-0.23, 0.63]$.

This enables computing conditional probabilities and by applying the transformation, we obtain conditional possibilities for this attribute given the class 'Set' of the first instance: $P(I_{SL=0.63}|Set) = P(-0.23 < SL < 0.63) = \int_{-0.23}^{0.63} g(SL, 0.2, 0.258)d_{SL}$

$P(I_{SL=0.63}|Set) = G(\frac{0.63-0.2}{0.258}) - G(\frac{-0.23-0.2}{0.258}) = 2 * G(\frac{0.43}{0.258}) - 1$

Since $G(1.666) = 0.9515$, $P(I_{(SL=0.63)}|Set) = 0.903$

$\pi(SL = 0.63|c_{j1} = Set) = 1 - P(I_{(SL=0.63)}|Set) = 1 - 0.903 = 0.097$

Table 5.6 show conditional possibility distributions of different attributes for each instance k calculated in the same manner as for the attribute SL.

Table 5.6: Individual possibility distribution for each kernel k

a_k	$\pi(SL = 0.63\|c_{jk})$	$\pi(SW = 0.79\|c_{jk})$	$\pi(PL = 0.9\|c_{jk})$	$\pi(PW = 0.81\|c_{jk})$
1	**0.097**	0.606	0.0007	0.003
2	0.048	0.330	0.0007	0.005
3	0.022	0.705	0.0005	0.003
4	0.015	0.499	0.0009	0.013
5	0.069	0.416	0.0007	0.002
6	0.373	0.705	0.494	0.299
7	0.727	0.705	0.396	0.380
8	0.261	0.499	0.604	0.380
9	0.296	0.002	0.207	0.230
10	0.615	0.116	0.443	0.380
11	0.846	0.936	0.699	0.462
12	0.561	0.062	0.723	0.816
13	0.152	0.330	0.760	0.928
14	0.846	0.203	0.954	0.694
15	0.261	0.330	0.824	0.801

Conditional possibility distributions for each individual kernel k allow to deduce the final possibilistic kernels of attribute SL for each class by applying the average as follows using (5.7):

$$\pi(SL = 0.63|Set) = (0.097 + 0.048 + 0.022 + 0.015 + 0.069)/5 = 0.050$$

$$\pi(SL = 0.63|Col) = (0.373 + 0.727 + 0.261 + 0.296 + 0.615)/5 = 0.454$$

$$\pi(SL = 0.63|Gin) = (0.846 + 0.561 + 0.152 + 0.846 + 0.261)/15 = 0.533$$

Table 5.7 includes the calculus results of the final conditional possibility distributions of the remaining attributes.

Table 5.7: Conditional possibility distribution of attributes for the FNPC

	$\pi(SL = 0.63\|c_j)$	$\pi(SW = 0.79\|c_j)$	$\pi(PL = 0.9\|c_j)$	$\pi(PW = 0.81\|c_j)$	$\pi(c_j\|a)$
Set	0.050	0.511	0.0007	0.005	8.9 E-08 (7.6 E-07)
Col	0.454	0.405	0.429	0.333	0.0262 (0.226)
Gin	0.533	0.372	0.792	0.740	**0.1162 (1)**

Using the product operator, we estimate the posterior possibility distributions of classes given the instance a:

$$\Pi(Set|a) = \Pi(SL|Set) * \Pi(SW|Set) * \Pi(PL|Set) * \Pi(PW|Set)$$

$= 0.050 * 0.511 * 0.0007 * 0.005 = 8.9E-08$

$\Pi(Col|a) = 0.454 * 0.406 * 0.429 * 0.334 = 0.0264$

$\Pi(Gin|a) = 0.533 * 0.372 * 0.792 * 0.740 = 0.1164$

Using normalization (last column in Table 5.7), $c^* = arg\,max_{cj}(7.6E{-}07, 0.226, \mathbf{1})$

As in the case of the NPC (Example 5.1), the instance a will be assigned to the class "$IrisVirginica$". However, we note that posterior possibility distributions of classes $\pi(c_j|a)$ in the case of the $FNPC$, are more adjusted than those obtained in the case of the NPC. This is can be explained by the fact that possibility distributions for the $FNPC$ are obtained using a *deep* data representation thorough examining individual possibility distributions $\pi(a_i|c_{jk})$ calculated for each kernel k. For this approach, estimating individual possibilities $\pi(a_i|c_{jk})$ for each kernel k and then considering their averages enables producing more refined possibility distributions which are a closer representation of data.

Computing individual possibilistic kernels have an additional complexity and time consuming for the $FNPC$. This contributes to improve the efficiency of this classifier if compared to the NPC when confronted to real databases as will be proved in the experimentation chapter.

5.4.2 Approximate Equality-based Interpretations of Data

In this section we propose two other methods for building a possibility distribution directly from a set of data, without computing a Gaussian probability distribution first. This type of approach is well in agreement with the generalized set-like view of possibility distributions, as previously pointed out. Indeed, a possibility can take into account the similarity between an observed value of an attribute and other observed values of the same attribute in the training examples. From a logical point of view, one can assume that $\Pi(a_i|c_j) = 1$ as soon as the value a_i has been observed at least one time in association with the class c_j. Conversely, if a value a'_i has not been observed in association with the class c_j it does not necessarily mean that $\Pi(a'_i|c_j) = 0$. In such case, we may consider that $\Pi(a'_i|c_j)$ should all the closer to 1 as a'_i is closer to an observed value a_i. This non-frequentist idea was first suggested in [68]. It is worth emphasizing that this is a purely local view of the building of the distribution, which does not make any assumption on its shape. This type of approach still makes an independent assumption of the attribute with respect to the class.

In this framework, the two suggested classification methods use an *approximate equality relation* between numerical values. Let d be the distance between the two

values, this fuzzy relation, namely $\mu_E(d(x,y))$ estimates to what extent x is close to y as follows (in other words E is a fuzzy set with decreasing membership function on $[0,1]$ with a bounded support and such that $\mu_E(0)=1$):

$$\mu_E(d) = max(0, min(1, \frac{\alpha+\beta-d}{\beta})), \alpha \geq 0; \beta > 0. \tag{5.10}$$

This relation is parameterized by α and β. The parameters α and β are respectively fixed to 0 and 1 in (5.10) for simplicity, once d is normalized in [0, 1], for all attributes. $\alpha = 0$ means that we use a triangular membership function, while $\beta = 1$ means that $\mu_E(d) = 0$ only for the most distant values of attributes. This closeness relation is now used to build a fuzzy histogram from the data.

The Fuzzy Histogram Classifier (FuHC)

Namely, we use the fuzzy relation E to build a fuzzy histogram [161] for attribute a_i given a class c_j, in the following way:

$$\pi(a_i|c_j) = \frac{1}{N_j} \sum_{k=1}^{N_j} \mu_E(d(a_i, a_{ik})), \tag{5.11}$$

where N_j is the number of instances belonging to the class c_j. The idea is here to be more faithful to the way the data are distributed (rather than assuming a normal distribution), and to take advantage of the approximate equality for obtaining a smooth distribution on the numerical domain, and may be supplying the scarcity of data. In that respect, the parameters of the approximate equality relation, depending on their values, not only reflect the expression of a tolerance on values that are not significantly different for a given attribute, but may also express a form of extrapolation from the observed data. The distribution 5.11 can then be directly used in the classification procedure. The algorithm based on this method will be named *Fuzzy Histogram Classifier* (FuHC) in the following. This classifier is a generalized case of a previously proposed classification method for continuous data [21]. In [21], the authors exploited a reduced version of the proximity equality function defined in formula (5.10) and they used a distance metric applied to non normalized attributes.

Example 5.3.

This example presents a detailed calculus of conditional and posterior possibility distributions in the case of $FuHC$. For continuity reasons, we preserve the same training set of Table 5.2 and also the same instance to classify (Table 5.4).

Table 5.8 illustrates individual proximity measures between attribute values of the instance a to classify and those of each training instance k. To guarantee a proximity function with significant values, i.e. $0 < \mu_E(d(x,y)) < 1$, α and β are respectively fixed to 0 and 1, once distance d was normalized in [0,1], for all attributes.

Table 5.8: Results of individual proximity measures

a_k	μ_{1k} $= 1- \mid a_1 - a_{1k} \mid$	μ_{2k} $= 1- \mid a_2 - a_{2k} \mid$	μ_{3k} $= 1- \mid a_3 - a_{3k} \mid$	μ_{4k} $= 1- \mid a_4 - a_{4k} \mid$
1	=1-|0.63-0.2| = 0.57	=1-|0.79-0.92|= 0.87	=1-|0.9-0.02| = 0.12	=1-|0.81-0.04| = 0.23
2	0,49	0,75	0,12	0,27
3	0,41	0,90	0,10	0,23
4	0,37	0,83	0,14	0,36
5	0,53	0,79	0,12	0,19
6	0,77	0,90	0,82	0,73
7	0,91	0,90	0,78	0,77
8	0,71	0,83	0,87	0,77
9	0,73	0,21	0,67	0,69
10	0,87	0,59	0,80	0,77
11	0,95	0,98	0,90	0,81
12	0,85	0,52	0,91	0,94
13	0,63	0,75	0,92	0,98
14	0,95	0,67	0,99	0,90
15	0,71	0,75	0,94	0,94

To estimate conditional possibility distribution for each attribute and each class one can use the average of instance proximities belonging to that class given in Table 5.8. If we consider the first attribute SL, conditional possibility distribution for each class is computed by:

$\Pi(SL = 0.63|Set) = \mu_1(Set) = \sum_{k=1}^{5} \mu_{1k}/5$

$= (0.57 + 0.49 + 0.41 + 0,37 + 0.53)/5 = 0.47$

$\Pi(SL = 0.63|Col) = \mu_1(Col) = \sum_{k=6}^{10} \mu_{1k}/5$

$= (0.77 + 0.91 + 0.71 + 0.73 + 0.87)/5 = 0.80$

$\Pi(SL = 0.63|Gin) = \mu_1(Gin) = \sum_{k=11}^{15} \mu_{1k}/5$

$= (0.95 + 0.85 + 0.63 + 0.95 + 0.71)/5 = 0.82$

Table 5.9 includes the calculus results of conditional possibilities of each attribute a_i given the class c_j.

Table 5.9: Possibility distribution of each attribute given classes for the FuHC

	$\pi(SL = 0.63\|c_j)$	$\pi(SW = 0.79\|c_j)$	$\pi(PL = 0.9\|c_j)$	$\pi(PW = 0.81\|c_j)$	$\pi(c_j\|a)$
Set	0.47	0.83	0.12	0.26	0.012 (0.024)
Col	0.80	0.69	0.79	0.75	0.327 (0.646)
Gin	0.82	0.73	0.93	0.91	**0.506 (1)**

Finally posterior possibility distribution of classes can be deduced using the product operator:

$\Pi(Set|a) = \Pi(a_1|Set) * \Pi(a_2|Set) * \Pi(a_3|Set) * \Pi(a_4|Set)$

$= 0.47 * 0.83 * 0.12 * 0.26 = 0.012$

$\Pi(Col|a) = \Pi(a_1|Col) * \Pi(a_2|Col) * \Pi(a_3|Col) * \Pi(a_4|Col)$

$= 0.8 * 0.69 * 0.79 * 0.75 = 0.327$

$\Pi(Gin|a) = \Pi(a_1|Gin) * \Pi(a_2|Gin) * \Pi(a_3|Gin) * \Pi(a_4|Gin)$

$= 0.82 * 0.73 * 0.93 * 0.91 = 0.506$

Thus $c^*= arg\ max_{cj}(0.027, 0.646, \mathbf{1})$

So the class $'IrisVirginica'$ will be assigned to the instance a since it is the most plausible.

Nearest Neighbor-based Possibilistic Classifier (NNPC)

We propose a second approach, named Nearest Neighbor-based Possibilistic Classifier (NNPC), which is based only on the analysis of the proximities between the attribute values a_{ik} belonging to each class c_j without counting them. The main idea of this classifier is to search for the nearest neighbor attribute value a_{ik} for the attribute value a_i of the item to be classified, in the training set of each class. The approximate equality function calculated between a_i and its nearest neighbor a_{ik} is then used to estimate the possibility distribution of the attribute value a_i given a class c_j as follows:

$$\pi(a_i|c_j) = \max_{k=1}^{N_j} \mu_E(d(a_i, a_{ik})). \tag{5.12}$$

In this approach, the closer an attribute value a_i to other attribute values of

instances belonging to a class c_j, the greater the possibility to belong to the class (w.r.t. the considered attribute). The expression (5.10 may be considered as a genuine possibility distribution [162]. An attribute value having a possibility 0 means that this value is not compatible with the associated class (it is the case when the value is not close to any other observed value of the attribute for the class). If the possibility is equal or close to 1, then the value is relevant for describing the class (a value having a small distance to instances of a class is considered as a possible candidate value in the representation of the class for a considered attribute).

Example 5.4.

Individual proximities given in Table 5.8 will be used here to estimate the conditional possibility distributions for the $NNPC$ as in the case of the $FuHC$; except that in this case, we estimate the conditional possibility of each attribute a_i given the class c_j by the *maximum* proximity among all proximities computed for each training instance k belonging to this class. This means that for this classifier, we only consider the attribute of the training instance with the highest proximity (the *nearest one*) without considering any one other, whereas in the case of the FuHC by taking the average, we are looking for all neighbors when estimating conditional distribution. Let us consider the attribute SL, the conditional possibility distribution for each class is as follow:

$\Pi(SL = 0.63|Set) = \mu_1(Set) = Max(0.57, 0.49, 0.41, 0.37, 0.53) = 0.57$

$\Pi(SL = 0.63|Col) = \mu_1(Col) = Max(0.77, 0.91, 0.71, 0.73, 0.87) = 0.91$

$\Pi(SL = 0.63|Gin) = \mu_1(Gin) = Max(0.95, 0.85, 0.63, 0.95, 0.71) = 0.95$

Table 5.10 summarizes calculus results of conditional possibilities of each attribute a_i given classes.

Table 5.10: Possibility distribution of each attribute given classes for the NNPC

	$\pi(SL = 0.63\|c_j)$	$\pi(SW = 0.79\|c_j)$	$\pi(PL = 0.9\|c_j)$	$\pi(PW = 0.81\|c_j)$	$\pi(c_j\|a)$
Set	0,57	0,90	0,14	0,36	0,026 (0,02)
Col	0,91	0,90	0,87	0,77	0,548 (0,608)
Gin	0,95	0,98	0,99	0,98	**0,9 (1)**

Thus we obtain the following posterior possibility measures :

$\Pi(Set|a) = \Pi(a_1|Set) * \Pi(a_2|Set) * \Pi(a_3|Set) * \Pi(a_4|Set)$

$= 0.57 * 0.90 * 0.14 * 0.36 = 0.026$

$\Pi(Col|a) = \Pi(a_1|Col) * \Pi(a_2|Col) * \Pi(a_3|Col) * \Pi(a_4|Col)$

$= 0.91 * 0.90 * 0.87 * 0.77 = 0.548$

$\Pi(Gin|a) = \Pi(a_1|Gin) * \Pi(a_2|Gin) * \Pi(a_3|Gin) * \Pi(a_4|Gin)$

$= 0.95 * 0.98 * 0.99 * 0.98 = 0.903$

$c^*= arg\ max_{cj}(0.028, 0.607, \mathbf{1})$

We can see that the class $'IrisVirginica'$ is always the class with the highest possibility so it is the most relevant class to assign to the instance a. Although classification results are equivalent for the four classifiers given in Example 5.1, 5.2 ,5.3 and 5.4, the estimated possibility distribution of classes is different. In particular, we note that Gaussian based Possibilistic classifiers distinguishes more between classes when generating possibility distributions whereas the proximity based Possibilistic classifiers produces possibility distributions that are more equitably distributed over the interval [0,1] which may cause conflict between classes for some particular cases.

However, high distinction between classes in the case of probability based possibility distributions cannot usually be justified. In fact, combining Gaussian distributions, having very small values, with the product operator may reinforce this "erroneous" distinction between classes. This conflicting problem will be discussed in the next section.

5.4.3 Detecting ambiguities in possibilistic classifiers as a basis for improvement

In some cases, classes may have very close plausibility estimates. In [23], we have proposed a multiple classification approach to deal with such conflicting situations. Instead of classifying a new instance in the most plausible class, the idea is to consider more than one class at a time when the plausibility difference between the most relevant classes is negligible. Experimental results for the multiple-classification approach showed that, for all data sets, classification accuracy of NPC and NNPC was significantly increased in the case of multiple-classification by comparison with the classical classification. Besides, the accuracy of NBC was not really increased by a similar procedure because the probability rates were generally significantly different. This is due to the use of product and division applied to numbers that are generally small. On the basis of these preliminary results, one may expect that possibilistic classifiers will have a better ability to detect confusion between classes than Bayesian ones. In this section, we first discuss how to evaluate ambiguity in classifiers, and how to compare possibilistic and Bayesian classifiers in terms of their ability to distinguish between classes. Then, we propose a method to improve the performance

of possibilistic classifiers on the basis of the detected ambiguities.

Meaningfulness of ambiguity in possibilistic classification

The intuitive idea behind this study is to estimate to what extent classification error is related to the ambiguity between close plausibility evaluations. In order to do that, we first define the classification ambiguity for an instance a with respect to classes as follows [24]:

$$AmbiguityDiff(a, c_1, ..., c_n) = 1 - (\Pi(c_1|a) - \Pi(c_2|a)) \tag{5.13}$$

where c_1 and c_2 are respectively the most and the second most relevant classes for a.

As experimentally checked, such a difference-based ambiguity measure is not appropriate for comparing probability values. Indeed, since these values are obtained as products (and quotient) of usually small values, fixing a threshold that is independent from the data set is not possible in general. For this reason we use a more appropriate ambiguity measure for Bayesian classifiers based on the ratio of probability of the second most relevant class and the first most relevant class:

$$AmbiguityRatio(a, c_1, ..., c_n) = \frac{P(c_2|a)}{P(c_1|a))} \tag{5.14}$$

The Hybrid Possibilistic Classification approach (HPC)

In classification problems, the main issue is to derive a model which is able to predict a unique class for any unseen example. Assigning a unique class to an example in a justified way may become difficult when the available information provided by the training examples is incomplete. This information may be incomplete in two respects. First, the training provides only an incomplete sampling which may be very scarce in some areas of the attribute space. Besides, the attribute vocabulary may be insufficient for discerning between examples having close descriptions but belonging to different classes. Regardless to the learning method, it may seem debatable to overcome such lacks of information and still justifying a unique classification. However, another limitation of the discriminating power of classifier may come from systematic independence assumption, as done naive Bayesian-like classifiers (probabilistic or possibilistic). If we are able to detect when the classification of an example may be problematic, this kind of limitation may be, at least partially, overcome. The idea is to take advantage of the fact that Bayesian-like classifiers

allows for an ambiguity analysis based on the plausibility degrees of belonging to a class. Then, problematic classifications may be detected, and in this case, a second algorithm (which does not make the independent assumption) can be applied for breaking the ties.

Thus, we propose to exploit a *Hybrid Possibilistic Classification* (HPC) approach [24] which aims at improving the accuracy of each of the previously introduced possibilistic classifiers. In this scope, we combine each proposed classifier with a Nearest Neighbour Heuristic (NNH). The nearest neighbour based classification is a local method that classifies an example on the basis of its similarity with the training examples in its neighbourhood. In our context, NNH has two advantages: i) its less sensitivity to the violation of the independence assumption, ii) due to the local nature of NNH an additional computer time cost only occurs in case of ambiguity. We expect that this heuristic may help the Bayesian-like classifiers to choose between classes whose plausibility are too close by preferring one on a nearest neighbour basis, instead of just choosing the most plausible class, even if the plausibility difference is not significant.

We consider that a classifier is in a failure state if the ambiguity (defined by 5.13 or 5.14) overcomes some fixed threshold ε. Note that, having a too liberal threshold would amount to use only the NNH. The HPC is detailed in the following algorithm:

Algorithm 9 Hybrid Possibilistic Classification Algorithm (PC)

Select an instance a_{ts} to classify
Class$(a_{ts}) \leftarrow \emptyset$
Classify a_{ts} by PC
$c_1 \leftarrow$ Most plausible class
$c_2 \leftarrow$ Second most plausible class
if $Ambiguity(a_{ts}, c_1, c_2) < \varepsilon$ **then**
  Class$(a_{ts}) \leftarrow c_1$
else
  $\pi(Instance1|a_{ts}) \leftarrow NNH(a_{ts}, c_1)$
  $\pi(Instance2|a_{ts}) \leftarrow NNH(a_{ts}, c_2)$
  if $\pi(Instance1|a_{ts}) > \pi(Instance2|a_{ts})$ **then**
    Class$(a_{ts}) \leftarrow c_1$
  else
    Class$(a_{ts}) \leftarrow c_2$
  end if
end if
return Class(a_{ts})

Given an instance a_{ts} to be classified, for each training instance a_{tr} labeled with the class c_j, the NNH estimates the possibility degrees $\pi(a_{tr}|a_{ts})$ as follows:

$$\Pi(a_{tr}|a_{ts}) = \Pi(a_1|a_{tr}) * ... * \Pi(a_M|a_{tr}) \tag{5.15}$$

With:

$$\pi(a_i|a_{tr}) = \mu_E(d(a_{i_{ts}}, a_{i_{tr}})) \tag{5.16}$$

The * may be the *minimum*, or the *product*. We may also think of using the leximin refinement of the minimum that amounts, when comparing two vectors of evaluations, to first reorder them increasingly, and then to reduce the comparison to a minimum-based evaluation of the subvectors made of the values where the two reordered vectors are not (approximately) equal. The attribute $a_{i_{ts}}$ (respectively $a_{i_{tr}}$) is the attribute of level i of the instance a_{ts} (respectively a_{tr}).

The nearest neighbour training instance to a_{ts} for the class c_j is the instance having the highest possibility among all instances a_{tr} belonging to the training set labeled with c_j.

$$a^*_{tr} = arg \max_{a_{tr}} \pi(a_{tr}|a_{ts}) \tag{5.17}$$

5.4.4 Computing possibility distribution as a family of Gaussian distribution from a sample set of data

In this section, we propose a novel way to build a possibility distribution from a set of data. Contrary to other probability to possibility based classifiers previously presented in the beginning of this section, we suppose here that the data follows a Gaussian distribution with *unknown parameters*. By taking into account the uncertainty attached to the estimation of these parameters from a sample set, we propose to build the possibility distribution that encodes all the Gaussian distributions that may have generated the data with a chosen confidence level. Then, we extend this approach to Gaussian kernels. The classifiers that we develop in this section are considered as a variant of those previously proposed in [23] [24].

Confidence region of the normal distribution parameters

Suppose that we have n observations $X_1, X_2, \cdots X_n$ drawn from a normal distribution with unknown mean μ and unknown variance σ^2. The $1-\alpha$ confidence region for the parameters of $\mathcal{N}(\mu, \sigma^2)$, contains a region in the two dimensional space of μ and σ^2 which has a probability of $1-\alpha$ to contain the true parameters value μ

and σ^2. Arnold and Shavelle in [7] have compared several methods for finding such confidence regions. In their paper, they present the method that we describe below and they call it the Mood's method. The idea of Mood confidence region is to take α_1 and α_2 such as $1-\alpha = (1-\alpha_1)(1-\alpha_2)$, where $1-\alpha$ is the confidence level of the found region. Considering $\overline{X} = \frac{X_1+\cdots+X_n}{n}$ and $S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2}$ respectively as the mean and the standard deviation estimated on the sample set, the confidence region $\mathcal{R}(n, \overline{X}, S)$ is defined by:

$$\mathcal{R}(n, \overline{X}, S) = \{(\mu, \sigma^2) : \frac{n-1}{\chi^2_{1-\frac{\alpha_2}{2},n-1}} S^2 \leq \sigma^2 \leq \frac{n-1}{\chi^2_{\frac{\alpha_2}{2},n-1}} S^2, \tag{5.18}$$

$$\overline{X} - \Phi_{1-\frac{\alpha_1}{2}} \frac{\sigma}{\sqrt{n}} \leq \mu \leq \overline{X} + \Phi_{1-\frac{\alpha_1}{2}} \frac{\sigma}{\sqrt{n}}\}. \tag{5.19}$$

Where Φ_q and $\chi_{q,k}$ are respectively the q^{th} quantile of the standard normal distribution and the q^{th} quantile of the chi square distribution with k degree of freedom. The authors also provide a table that indicates the optimal combination of α_1 and α_2 that gives the smallest possible region for a fixed confidence level $1-\alpha$ and for a fixed number of observations n.

By using the equations (5.18) and (5.19) we can find the mean and variance confidence interval respectively $[\mu_{min}, \mu_{max}]$,$[\sigma^2_{min}, \sigma^2_{max}]$, associated with our confidence region. Once we have found the confidence region, we define Θ as the family which contains all the probability functions p in the confidence region i.e.

$$\Theta = \{p = \mathcal{N}(\mu, \sigma^2) | (\mu, \sigma^2) \in \mathcal{R}(n, \overline{X}, S)\}.$$

Possibility distribution for a family of Gaussian distribution

We have shown how to build a confidence region for the parameters of a normal distribution (for a simplification purpose, we always take $1-\alpha = 0.95$ for the regions in the following). If the estimation of these parameters is a critical issue of a decision process, it may be interesting to take into account the uncertainty around the parameters of the normal distribution that may have generated the data. In this scope, we propose to construct the most specific possibility distribution that contains the family Θ of Gaussian distributions that have mean and variance parameters in the confidence region.

We name $\Lambda = \{\pi | \pi = Tr(p), p \in \Theta\}$ the set of possibility distribution obtained by transforming each distribution in Θ ($Tr(p)$ is the possibility transformation of a probability distribution using Formula 1.23). Thus, the possibility distribution defined by

$$\pi_{(n,\overline{X},S)}(x) = sup\{\pi(x) | \pi \in \Lambda\}$$

encodes all the family Θ. $\pi_{(n,\overline{X},S)}$ has the following definition:

$$\pi_{(n,\overline{X},S)}(x) = \begin{cases} 1 \text{ if } x \in [\mu_{min}, \mu_{max}] \\ 2 * G(x, \mu_{min}, \sigma^2_{max}) & \text{if } x < \mu_{min} \\ 2 * G(2 * \mu_{max} - x, \mu_{max}, \sigma^2_{max}) & x > \mu_{max} \end{cases} \tag{5.20}$$

where μ_{min}, μ_{max} and σ^2_{max} are respectively the lower and the upper bounds of the mean confidence interval, and the upper bound of the variance confidence interval associated to the confidence region found by (5.19) and (5.18). Moreover, $\mathcal{G}(x, \mu, \sigma^2)$ is the cumulated distribution function of the $\mathcal{N}(\mu, \sigma^2)$. For a more detailed discussion, see [6].

Figure 5.1 presents the distributions $\pi_{(10,\overline{X},S)}$ for the family of Gaussian distribution (in gray) that are in the mood region obtained from a sample of 10 pieces of data that follows the distribution $\mathcal{N}(0, 1)$.

Probability to possibility transformation-based classifiers

We apply the method presented above to Naive Bayesian classifiers, where the distributions are assumed to be normal, and then to its flexible extension FNBC (using a combination of normal distributions). We shall call NPC-2 and FNPC-2 [20] the possibilistic extensions of NBC and FNBC. In the possibilistic setting, we still assume that the probability distributions we start with are normal (or a combination of normal distributions), but we also encode the uncertainty around the estimations of their parameters.

In order to build NPC-2, we need to compute three types of possibility degrees: $\pi(c_i)$ the possibility of a class c_i, $\pi(a_i)$ the possibility of the attribute value a_i, and $\pi(a_i|c_j)$ the conditional possibility of a_i knowing c_j. These values are obtained as follows:

- $\pi(c_i)$ is obtained by computing the probability-possibility transformation (using equation 1.22) of the prior probability distribution over the classes;
- $\pi(a_i)$ is obtained by computing (using equation 5.20) the possibility distribution $\pi_{(N,\overline{X}_i,S_i)}$ that encodes the confidence region $\mathcal{R}_i(N, \overline{X}_i, S_i)$ for the parameters of the normal distributions of A_i where N is the number of examples in the database, $\overline{X}_i$ is the means of the a_i values and S_i their standard deviation;
- $\pi(a_i|c_j)$ is obtained by computing (using equation 5.20) the possibility distribution $\pi_{(N_j,\overline{X}_{(i,j)},S_{(i,j)})}$ that encodes the confidence region $\mathcal{R}_{(i,j)}(N_j, \overline{X}_{(i,j)}, S_{(i,j)})$ for the parameters of the normal distributions of A_i where N_j is the number of

examples in the database that are associated to the class c_j, $\overline{X}_{(i,j)}$ is the means of the a_i values on this subset and $S_{(i,j)}$ their standard deviation.

The FNPC-2 is exactly the same as the NPC-2 in all respects, except that the method used for density estimation on continuous attributes is different. Rather than using a single Gaussian distribution for estimating each continuous attribute, we use a kernel density estimation as in FNBC. Kernel estimation with Gaussian kernels looks much the same except that the estimated density is averaged over a large set of kernels. For the FNPC-2, we use the following expression:

$$\pi(a_i|c_j) = \frac{1}{N_j} \sum_{k=1}^{N_j} \pi_{(N_j,\mu_{ik},\sigma)}(a_i) \tag{5.21}$$

where a_i is a value for the attribute A_i, k ranges over the N_j instances of the training set in class c_j and $\mu_{ik} = a_{ik}$ (a_{ik} is the value of the attribute A_i for the k-th example in the considered subset). For all distributions, the standard deviation is estimated by

$$\sigma = \frac{1}{\sqrt{N}}.$$

5.5 Possibilistic distributions for imperfect numerical data

In many domains, databases are supplied with various information sources which may be neither fully reliable nor precise. That is why, available information is often pervaded with uncertainty and imprecision. In particular for numerical data, many factors contribute to make them imperfect, such as the variability of data, the use of unreliable data transmission or outdated sources, or the measurement errors. For instance, data provided by sensor networks such as temperature, pressure and rain measurement may be uncertain or imprecise.

In the context of classification or diagnosis problems, attributes in the training or testing sets may have uncertain numerical values. In practice, when classifying unseen examples, only an interval for a numerical value may be given instead of a precise value in some situations.

Imperfection can also affect categorical data and especially the training set labels. In fact, the framework of supervised learning which assumes precision and full certainty does not necessarily correspond to practical situations. The acquisition of a large volume of certain and precise labeled data may be problematic in some real

domains, for cost reasons or partial lack of knowledge on the problem to be solved. It is often the case when the data are labeled by experts.

This kind of labeling work is thus often costly; moreover, an expert may need to express uncertainty or imprecision in this task. Indeed the precise qualification of a situation by an expert may be difficult (e.g., in medical diagnosis, or in law application data).

Since classical classification techniques are inappropriate to deal with such imperfect data, two solutions are commonly considered: either ignoring such data by regarding them as unknown or incomplete, or developing suitable tools for treating them. In the first approach information is lost and this may lead to inaccurate classification models. On the contrary, if we adjust classification techniques in order to be able to deal with imperfect data, the models produced will describe the concepts more faithfully.

Various formalisms have been proposed to deal with imperfect data, in the different uncertainty settings. The development of solutions enabling the handling of imprecise attribute values or uncertain classes has particularly interested some authors during the last past years.

In this section, we extend possibilistic classifiers, proposed in [23] [24] and previously presented in Section 5.4, in order to handle uncertainty in data representation. Before proposing solutions to deal with uncertainty in both training and testing data sets, we state basis hypothesis concerning the structure of these data sets for uncertain possibilistic classifiers .

5.5.1 Structure of uncertain training and testing data sets

Uncertainty pervades attribute values in the testing instances and classes in the training instances. All uncertain possibilistic classifiers [25] [20], proposed in this section, are based on the following hypotheses:

- All training instances are assumed to have perfect (certain and precise) attribute values as in "classical" possibilistic classifiers [23] [24]
- All testing instances have imprecise attribute values modeled by intervals.
- The class of any training instance is represented through a possibility distribution over the class values thus reflecting uncertainty on the classification.

Let us consider a classification problem with 3 class labels (c_1, c_2 and c_3) and a standard training instance a with M certain and precise numerical attributes and assigned to the class c_1:

$$a = (a_1, a_2, ..., a_M, c_1)$$

When an expert is unable to give an exact class label for some observed example, he may represent his knowledge about the class associated with this example by means of a possibility distribution over the different possible class labels. The uncertain version of instance a_U is then represented by:

$$a_U = (a_1, a_2, ..., a_M, \pi_{c_1}, \pi_{c_2}, \pi_{c_3})$$

where π_{c_i} is the possibility degree for the class c_i that reflects the partial ignorance of the expert. For instance, the distribution $(1, 0.3, 0.7)$ expresses that the expert finds the instance fully compatible with the first class, less compatible with the second one and still less compatible with the third one. There are some other noticeable particular cases that can also be represented as such an uncertainly classified instance. Thus the distribution (1, 1, 0) represents pure imprecision over the first and the second class labels which are fully plausible whereas the third class is impossible. Besides, a distribution such as (1, 0, 0) coincides with an original, certainly classified instance for which, here, only the first class is possible while the others are completely rejected. Finally, the expert may also express his total ignorance about the instance by choosing the distribution (1, 1, 1) according to which all class labels are fully plausible.

Uncertainty in the testing set concerns attribute values and each test instance may include certain or uncertain attribute values. Since we are only interested in continuous data in this framework, the proposed model allows an expert to express his imprecise knowledge by means of an interval restricting the attribute value. Thus, for each imprecise attribute, the observed value is supposed to be the form of $I_i = [L_i, U_i]$ where L_i and U_i are respectively the lower and the upper bounds for the true attribute value a_i such that $L_i < a_i < U_i$.

For imprecise attribute values, the degree of ignorance about the real value is related to the *width* of the interval for this attribute. For example, in the case of an attribute with values in $[0, 1]$, an interval such as [0.399, 0.401] is a rather precise representation whereas the interval [0.1, 0.9] models an high ignorance. The total ignorance is for the interval [0, 1].

5.5.2 Processing of uncertain classes in the training set

In this section instead of an exact class label, for each instance we assign a possibility distribution on the different possible labels. Our problem is to estimate a possibility distribution for each attribute a_i given the class c_j which can be the most specific representation for *uncertain* numerical data [25]:

$$\pi(a_i|c_j) = \frac{\pi(a_i, c_j)}{\pi(c_j)} \tag{5.22}$$

To combine possibility distributions over the training instances belonging to a specific class, one can exploit the *mean* operator (Formula 5.7 or 5.11) as in the perfect case. We extend the FNPC, FNPC-2 and FuHC as follows:

$$\pi(a_i, c_j) = \frac{1}{N}\sum_{k=1}^{N}\pi(a_i, c_{jk}) = \frac{1}{N}\sum_{k=1}^{N}\pi(a_i|c_{jk}) * \pi(c_{jk}) \tag{5.23}$$

For the NNPC, the *max* operator is used instead of the *mean*:

$$\pi(a_i, c_j) = \max_{k=1}^{N}\pi(a_i, c_{jk}) = \max_{k=1}^{N}\pi(a_i|c_{jk}) * \pi(c_{jk}) \tag{5.24}$$

Where $\pi(c_{jk})$ represents the individual possibility of the class c_j for each training instance k.

We note that the proposed model, supporting uncertainty in the class labels, also includes the certain case where $\pi(c_{jk})$ is 1 for the true label and 0 otherwise.

For the NPC-2 [20], possibility distribution $\pi(a_i|c_j)$ is obtained by computing (using equation 5.20) the possibility distribution $\pi_{(N_j,\overline{X}_{(i,j)},S_{(i,j)})}$ that encodes the confidence region $\mathcal{R}_{(i,j)}(N_j, \overline{X}_{(i,j)}, S_{(i,j)})$ for the parameters of the normal distributions of A_i. Since the class is now pervaded with uncertainty, we will use weighted sums for evaluating the values N_j, $\overline{X}_{(i,j)}$ and $S_{(i,j)}$. Then we have:

$$N_j = \sum_{k=1}^{N}\pi(c_{jk}),$$

$$\overline{X}_{(i,j)} = \frac{\sum_{k=1}^{N}\pi(c_{jk}) * a_{ik}}{N_j}$$

and

$$S_{(i,j)} = \frac{\sum_{k=1}^{N}\pi(c_{jk}) * (a_{ik} - \overline{X}_{(i,j)})^2}{N_j}.$$

Example 5.5.

Let us use a modified version of the training set T_r given in Table 5.2 page 123 to illustrate how to estimate conditional and posterior possibility distributions in presence of uncertain classes in this training set. For simplicity, we choose to deal only with the FuHC in this example. We have artificially included uncertainty to the training set T_r in the following manner: we suppose that for each instance in T_r is

Table 5.11: Uncertain Training set with normalized attributes

					$\pi(c_j \mid a_{U_k})$		
a_{U_k}	SL	SW	PL	PW	Set	Col	Gin
1	0,20	0,92	0,02	0,04	1	0.40	0,85
2	0,12	0,54	0,02	0,08	1	1	0,61
3	0,04	0,69	0	0,04	1	0,09	0
4	0,00	0,62	0,04	0,17	1	0,32	0,8
5	0,16	1,00	0,02	0,00	1	0,86	0,46
6	0,40	0,69	0,72	0,52	0,87	1	0,6
7	0,72	0,69	0,68	0,57	0,2	1	0,95
8	0,92	0,62	0,77	0,57	0,59	1	1
9	0,36	0,00	0,57	0,48	0	1	0,21
10	0,76	0,38	0,70	0,57	0,74	1	0
11	0,68	0,77	1,00	1,00	0,58	0,43	1
12	0,48	0,31	0,81	0,74	0,11	0,12	1
13	1,00	0,54	0,98	0,83	0	0	1
14	0,68	0,46	0,91	0,70	0,76	1	1
15	0,92	0,54	0,96	0,87	0,62	0,02	1
				$\pi(c_j)$	0.63	0.62	0.7

given a possibility distribution over class values. The uncertain training set is given in Table 5.11.

In this example, we still consider the same certain test instance given by Table 5.4. Conditional possibility distributions for uncertain training data are estimated in the same manner as in the certain case (see Example 5.3 for the FuHC) except that, in the uncertain case, we should consider the individual possibility of each class for each instance since it is no longer 0/1.

Let us show a detailed computation of conditional possibilities for the first attribute SL. We first start by estimating individual possibilities for each instance and each class. Since each training instance is assigned with a degree of possibility, at the same time, to the three possible class values (Set, Col and Gin), each of these instances should be considered as a training instance for each class value when estimating the priori conditional distribution for attributes. Thus for each attribute and each class, one should compute 15 individual possibilities instead of 5 in the certain case. So for the first instance, the following three possibilities are computed:

$\pi(SL = 0.63, c_{j1} = Set) = \pi(SL = 0.63 | c_{j1} = Set) * \pi(Set_1) = (1 - |0.63 - 0.2|) * 1 = 0.57$

$\pi(SL = 0.63, c_{j1} = Col) = \pi(SL = 0.63 | c_{j1} = Col) * \pi(Col_1) = (1 - |0.63 - 0.2|) * 0.4 = 0.23$

$\pi(SL = 0.63, c_{j1} = Gin) = \pi(SL = 0.63|c_{j1} = Gin) * \pi(Gin_1) = (1 - |0.63 - 0.2|) * 0.85 = 0.48$

For the remaining instances and for other attributes, we should apply the same process as we did for the first instance and the first attribute. Results of individual possibility distribution for all attributes, instances and classes are given in Table 5.12.

Table 5.12: Results of individual possibility distribution of attributes for the FuHC

	$\pi(SL = 0.63, c_{jk})$			$\pi(SW = 0.79, c_{jk})$			$\pi(PL = 0.9, c_{jk})$			$\pi(PW = 0.81, c_{jk})$		
a_{U_k}	**Set**	**Col**	**Gin**	**Set**	**Col**	**Gin**	**Set**	**Col**	**Gin**	**Set**	**Col**	**Gin**
1	0,57	0,23	0,48	0,87	0,35	0,74	0,30	0,12	0,26	0,39	0,16	0,33
2	0,49	0,49	0,30	0,75	0,75	0,46	0,22	0,22	0,13	0,31	0,31	0,19
3	0,41	0,04	0,00	0,90	0,08	0,00	0,14	0,01	0,00	0,23	0,02	0,00
4	0,37	0,12	0,30	0,83	0,26	0,66	0,10	0,03	0,08	0,19	0,06	0,15
5	0,53	0,46	0,24	0,79	0,68	0,36	0,26	0,22	0,12	0,35	0,30	0,16
6	0,67	0,77	0,46	0,79	0,90	0,54	0,44	0,50	0,30	0,51	0,59	0,35
7	0,18	0,91	0,86	0,18	0,90	0,86	0,16	0,82	0,78	0,18	0,91	0,86
8	0,42	0,71	0,71	0,49	0,83	0,83	0,58	0,98	0,98	0,53	0,89	0,89
9	0,00	0,73	0,15	0,00	0,21	0,04	0,00	0,46	0,10	0,00	0,55	0,12
10	0,64	0,87	0,00	0,44	0,59	0,00	0,64	0,86	0,00	0,70	0,95	0,00
11	0,55	0,41	0,95	0,57	0,42	0,98	0,45	0,34	0,78	0,50	0,37	0,87
12	0,09	0,10	0,85	0,06	0,06	0,52	0,06	0,07	0,58	0,07	0,08	0,67
13	0,00	0,00	0,63	0,00	0,00	0,75	0,00	0,00	0,90	0,00	0,00	0,81
14	0,72	0,95	0,95	0,51	0,67	0,67	0,59	0,78	0,78	0,66	0,87	0,87
15	0,44	0,01	0,71	0,46	0,01	0,75	0,61	0,02	0,98	0,55	0,02	0,89

Conditional possibility distribution are estimated by the averaged joined possibilities divided by the possibility of each class as in Equation (5.22). For example, $\pi(SL = 0.63|Set) = (0.57 + 0.49 + 0.41 + ... + 0.44)/(15 * 0.63) = 0.64$. Table 5.13 gives the averaged conditional possibilities of each attribute a_i given the class c_j computed over the fifteen instances.

Table 5.13: Averaged conditional possibility distribution of each attribute given classes for the FuHC

	$\pi(SL = 0.63|c_j)$	$\pi(SW = 0.79|c_j)$	$\pi(PL = 0.9|c_j)$	$\pi(PW = 0.81|c_j)$	$\pi(c_j|a)$
Set	0.64	0.80	0.48	0.55	0.085(0.497)
Col	0.73	0.73	0.59	0.66	0.128 (0.748)
Gin	0.72	0.78	0.64	0.68	**0.171 (1)**

Posterior possibility distributions of classes are computed as in the certain case except that, in this case prior possibility of classes is no longer equal to 1:

$\Pi(Set|a) = \Pi(Set) * \prod_{i=1}^{4} \Pi(a_i|Set)$

$= 0.63 * (0.64 * 0.80 * 0.48 * 0.55) = 0.085$

$\Pi(Col|a) = \Pi(Col) * \prod_{i=1}^{4} \Pi(a_i|Col)$

$= 0.62 * (0.73 * 0.73 * 0.59 * 0.66) = 0.128$

$\Pi(Gin|a) = \Pi(Gin) * \prod_{i=1}^{4} \Pi(a_i|Gin)$

$= 0.7 * (0.72 * 0.78 * 0.64 * 0.68) = 0.171$

Thus $c^*= arg\ max_{c_j}(0.497, 0.748, \mathbf{1})$

So the class $'IrisVirginica'$ will be assigned to the instance a.

5.5.3 Processing of imprecise attributes in the testing set

In the following we propose an algorithm for handling imprecision in attribute values in the testing set. Let us consider a function F which estimates conditional possibilities for attribute values in the perfect case. For each *observed* attribute value x_i, this function estimates $\pi_{(a_i|c_j)}(x_i)$. Knowing that the observed value of an attribute is no longer a fixed value in the domain of the attribute but rather an interval, the problem returns to estimate $\pi_{(a_i|c_j)}(I_i)$.

In order to handle the evaluation of interval possibilities, we use the extension principle [173] presented in Section 1.3.3 page 18. Let F be a real function such that $F : X \rightarrow R, R$ being the set of real numbers. Let $F(x) = u$ and let $\pi_F(u)$ be the possibility for u. If we apply the extension principle to possibility distribution, the possibility degree for u can be defined by:

$$\pi_F(u) = sup\{\pi(x)|F(x) = u\}. \tag{5.25}$$

Assume $I_1, ..., I_M$ are uncertain observations for attributes $a_1, ..., a_M$. To estimate the possibility distribution for an interval I_i, the equation (5.25) becomes:

$$\pi(I_i|c_j) = sup\{\pi(a_i|c_j), a_i \in I_i\} \tag{5.26}$$

To define conditional possibilities for each uncertain observation I_i of the testing instance, we consider the following algorithm:

1. Search for all attribute values a_i in the training set such that $a_i \in I_i$
2. Compute the possibility of attribute values a_i given the class c_j by equation 5.22
3. Consider the highest possibility to estimate the possibility of I_i

Example 5.6.

Let us consider an uncertain version of the test instance given in Table 5.14. This instance contains two certain (SL and PL) and two uncertain attributes (SW and PW). Uncertainty related to attributes is represented through an interval that covers the true value of each attribute:

Table 5.14: Uncertain instance to classify

	SL	**SW**	**PL**	**PW**
Instance: a_U	0.63	[0.6,0.81]	0.9	[0.72,0.86]

In this example, we are only interested to show the process of classifying *uncertain instances* in the test set. For this reason and in order to simplify the calculus process, we use the *certain* training set of Table 5.2 instead of that of Table 5.11. In the following, we present the process of estimating conditional possibility distribution for uncertain attributes (SW and PW). Possibilities for certain attributes (SL and PL) are computed as in Example 5.3 for the $FuHC$.

To classify the uncertain test instance, for each uncertain attribute (represented by an interval) one should look for all attribute values in the training set that belongs to this interval when estimating its conditional possibility distribution. So the first step is to search for all these attribute values and then estimate the conditional possibility distributions for each of them.

If we consider the uncertain attribute $SW = [0.6, 0.81]$, attribute values in the training set that are in this interval are $\{0.62, 0.69, 0.77\}$ (see Table 5.2 page 123). Similarly, attribute values belonging to the interval $PW = [0.72, 0.86]$ are $\{0.74, 0.83\}$.

To estimate conditional possibility for each attribute value in each interval (i.e. $\Pi(SW = 0.62|c_j)$, $\Pi(SW = 0.69|c_j)$, etc), one should follow the same process described in Example 5.3 (for the FuHC) which returns to compute individual possibility distributions for each training instance and then take the average to estimate the final conditional distribution for each attribute.

Table 5.15 (respectively Table 5.16) gives the averaged conditional possibility distribution for each attribute value SW (resp. PW) belonging to each interval. For

each class, the conditional possibility to be considered for intervals $SW = [0.6, 0.81]$ and PW=[0.72,0.86] corresponds to the maximum possibility over attribute values in each interval.

Table 5.15: Conditional possibility distribution of each SW attribute value for the FuHC

$\Pi(SW\|c_j)$	SW=0.62	SW=0.69	SW= 0.77	$\Pi(I_{SW=[0.6,0.81]}\|c_j)$ =Max
Set	0.83	**0.85**	0.83	0.85
Col	**0.80**	0.79	0.71	0.80
Gin	**0.84**	0.80	0.75	0.84

Table 5.16: Conditional possibility distribution of each PW attribute value for the FuHC

$\Pi(PW\|c_j)$	PW=0.72	PW=0.85	$\Pi(I_{PW=[0.72,0.86]}\|c_j)$ = Max
Set	**0.35**	0.22	0.35
Col	**0.84**	0.71	0.84
Gin	0.88	**0.91**	0.91

Finally we can estimate posterior possibility distributions of classes given the uncertain instance a_U. In the following, bold values corresponds to conditional possibilities for uncertain attributes given in Tables 5.15 and 5.16 whereas non bold values corresponds to certain ones taken from Table 5.9 (Example 5.3). From these results, we can see that the uncertain test instance will be assigned to the third class:

$\Pi(Set|a_U) = \Pi(SL|Set) * \Pi(I_{SW}|Set) * \Pi(PL|Set) * \Pi(I_{PW}|Set)$

$= 0.47 * \mathbf{0.85} * 0.12 * \mathbf{0.35} = 0.016(0.027)$

$\Pi(Col|a_U) = 0.8 * \mathbf{0.8} * 0.79 * \mathbf{0.84} = 0.424(0.728)$

$\Pi(Gin|a_U) = 0.82 * \mathbf{0.84} * 0.93 * \mathbf{0.91} = \mathbf{0.582}$ **(1)**

5.6 Conclusion

This work has investigated a possibilistic classification paradigm that may be viewed as a counterpart of Bayesian classification and that applies to continuous attribute domains. Then an important issue is the estimation of possibilistic distributions from numerical data, without discretization. For this purpose, we have proposed and tested the performance of two families of possibilistic classifiers: the first family

called, Gaussian-based Possibilistic Classifiers, assumes normality assumption when estimating possibilistic distributions. For this family of classifiers, we have used a probability-possibility transformation method enabling us to derive a possibilistic distribution from a probabilistic one. First, we have applied the transformation method to move from a classical NBC to a NPC, which introduces some further tolerance in the description of classes. Then, we have tested the feasibility of a Flexible Naive Possibilistic Classifier, which is the possibilistic counterpart of the Flexible Naive Bayesian Classifier. The FNPC estimates possibilistic distributions in a non-parametric way by applying the transformation method to kernel densities instead of Gaussian ones. The intuition behind this classifier is that kernel densities are less sensible than Gaussian ones to normality violation.

In the same context, we have studied a second variant of probability-possibility transform based classifiers. Here the possibility distributions that are used are supposed to encode the family of Gaussian probabilistic distributions with unknown parameters. First, we have applied the transformation method to move from a classical NBC to NPC-2, which takes into account the confidence intervals of the Gaussian distributions. Then, we have tested the feasibility of a Flexible Naive Possibilistic Classifier (FNPC-2), which is the possibilistic counterpart of the Flexible Naive Bayesian Classifier.

The second family of possibilistic classifiers abandons the normality assumption and has a direct representation of data. We have proposed two other classifiers named Fuzzy Histogram Classifier and Nearest Neighbor-based Possibilistic Classifier in this context. The two proposed classifiers exploit an idea of proximity between attribute values in order to estimate possibility distributions. In the first classifier, we compute an average proximity, whereas in the second one we analyze proximities between attributes without counting them. The main advantage of this family of classifiers, when compared to the first one, is their ability to derive possibilistic distributions without the need of the normality assumption, which may lead to a more realistic representation of data.

By studying possibilistic classifiers, we have noted that they have a higher ability to detect ambiguity between classes than Bayesian classifiers. Namely the former acknowledge the fact that it is difficult to classify some examples by assessing close possibility degrees to competing classes, whereas the latter in the same situation may give the illusion of discriminating between classes by assessing very different probability degrees to them.

As an attempt to improve the performance of possibilistic classifiers, we have proposed an hybrid classification method that is based on a Nearest Neighbor Heuristic used for separating classes having close plausibility estimates. The Nearest Neighbor Heuristic contributes to help the main classifier to converge to the correct class label in case data information is insufficient for a more precise classification, rather than

choosing between classes having very close plausibility estimates in a rather arbitrary way.

The second interest of this chapter is to extend the proposed possibilistic classifiers for handling uncertainty and imprecision in input data sets. Two types of uncertainty are considered: i) uncertainty related to class attribute in the training set modeled through possibility distributions over class labels and ii) uncertainty related to attribute values in the testing set represented through intervals for continuous data. For the first type of uncertainty, we have adapted the possibilistic classification model suitable for the certain case, to support uncertainty in class labels. We have also showed that the adjusted model is suitable for the perfect as well as the imperfect case.

We have also proposed an algorithm based on the extension principle to deal with uncertainty in the attribute values. This algorithm seeks to estimate possibility distributions for an uncertain attribute (interval) by looking for possibility distributions of each attribute in the training set belonging to this interval.

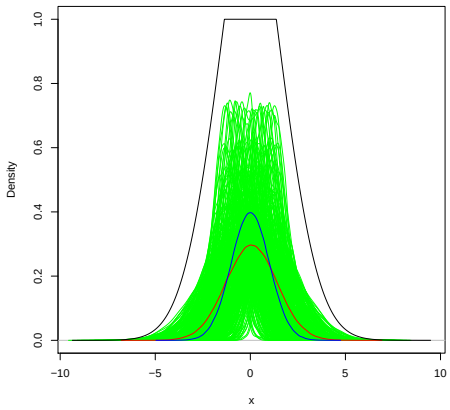

Figure 5.1: An example of the possibility distribution for the family Θ, with a confidence level of 0.95 and a dataset with n=10.

Chapter 6

Experimenting Possibilistic Classifiers for perfect/imperfect data

6.1 Introduction

In order to analyze the performance of possibilistic classifiers, either in the perfect or imperfect case, we implemented and tested all these classifiers as will be presented in the Appendix. This will allow us to conduct several experimental studies to show their advantages and also limits if compared to other Bayesian classifiers.

To test these possibilistic classifiers we use the same data sets used for experimenting Possibilistic rule-based classifiers in Chapter 4. The characteristics of these data sets are given in Table 4.4.

This chapter is organized as follows. In the next section we provide a exhaustive study for possibilistic classifiers when data sets are assumed to be perfect. This is an important step before dealing with imperfection because we aim to show to what extend possibilistic classifiers are efficient in the classification task. This section is divided in two parts: in the first one, we discuss classification results of possibilistic classifiers and compare them to Bayesian classifiers for all data sets. In the light of the results of this preliminary study, we carry out further experiments on extended versions of possibilistic classifiers in particular those based on computing possibility distributions as a family of Gaussian probability distributions. Section 3 includes results for possibilistic classifiers when we deal with imperfect data sets. In this section, we first give the way for artificially creating imperfect data sets. We also propose two classification accuracy measures usually used in such case. In the last part of this section we report results for the two types of uncertainty concerning classes and attributes and we provide a deep analysis for these results.

6.2 Experiments of possibilistic classifiers for the perfect numerical data

This section provides experimental results for the possibilistic classifiers that have been previously introduced when data is assumed to be perfect. In order to evaluate the accuracy of each classifier, we have used the standard *Percent of Correct Classification*(PCC) defined by Equation 2.9.

This experimental study is divided in two parts. First, we give a preliminary study of possibilistic classifiers for classifying numerical data. This study will enable us to experiment further extensions of these classifiers in the second part which leads to a significant improvement of the efficiency of possibilistic classifiers.

6.2.1 Preliminary study of Possibilistic Classifiers

In this subsection, we first evaluate possibilistic classifiers NPC, FNPC, FuHC and NNPC and compare our results to those of probabilistic ones, namely NBC and FNBC [107]. This comparative study is carried out through paired t-tests. Second, we compare the capabilities of possibilistic and probabilistic classifiers for detecting examples that are ambiguous with respect to classification. Third, we test the ability of the hybrid-classification method for improving the performance of the possibilistic classifiers. We use the signed-ranks test to measure the significance of this improvement.

Results and discussion for the Possibilistic Classifiers

We use the product in the aggregation step for all possibilistic classifiers, except for the NNPC where we use the minimum because it provides better results for the ambiguity study and it has been shown in [23] that the three versions (product, minimum, and a leximin-based refinement of minimum) have a competitive efficiency in this case. For the FuHC and NNPC, α and β are respectively fixed to 0 and 1 in equation 5.10 page 129 for simplicity, once d is normalized in [0, 1], for all attributes.

Table 6.1 shows the classification results obtained with NPC, NBC, FNPC, FNBC, FuHC and NNPC for the fifteen mentioned datasets. We also present those of the leximin based-NNH considered here as an independent classifier. By comparing the classification results of the first six classifiers we can notice that:

• For the two classifiers NPC and NBC, which assume that the attribute values are normally distributed, we remark that NPC is more accurate than NBC on four databases (Yeast, Ecoli, Glass and Heart) and less accurate on the remaining

Table 6.1: Experimental results given as the mean and the standard deviation of 10 cross-validations

	NPC	**NBC**	**FNPC**	**FNBC**	**FuHC**	**NNPC**	**NNH**
Iris	95.33±6.0 (4)	95.33±6.0 (4)	**96.0±5.33** (2)	95.33±5.21 (4)	94.66±4.0 (6)	90.66±4.42 (7)	**96.0±4.42** (2)
Cancer	95.03±2.26 (6)	96.34±0.97 (3)	97.37±1.82 (2)	**97.65±1.76** (1)	96.05±1.96 (5)	93.41±2.49 (7)	96.06±1.82 (4)
Wine	94.37±5.56 (4)	**97.15±2.86** (1)	96.6±3.73 (2.5)	96.67±5.67 (2.5)	93.26±4.14 (5.5)	92.64±5.12 (7)	93.26±5.98 (5.5)
Diabetes	69.01±3.99 (5)	74.34±4.44 (3)	**74.36±4.57 (1)**	74.35±3.38 (2)	73.44±5.31 (4)	67.96±6.05 (7)	67.97±5.73 (6)
Magic	59.24±7.09 (7)	66.02±5.37 (5)	73.37±2.96 (2)	72.8±3.29 (3)	68.34±6.69 (4)	64.80±2.41 (6)	**74.21±4.51 (1)**
Transfusion	61.67±6.6 (7)	72.6±4.56 (3)	67.43±7.43 (6)	70.09±7.68 (4)	72.76±7.19 (2)	**76.50±5.94 (1)**	68.73±5.61 (5)
Sat. Image	88.26±2.62 (6)	90.55±2.46 (4)	92.02±2.81 (3)	90.0±4.39 (5)	86.88±3.67 (7)	93.58±1.88 (2)	**93.95±2.6 (1)**
Segment	71.47±4.15 (7)	80.73±2.16 (6)	**90.73±1.8** (2.5)	88.27±3.19 (4)	81.07±3.51 (5)	90.73 (2.5)±2.15	**95.07±1.61 (1)**
Yeast	49.67±4.87 (5)	48.65±4.42 (6)	52.02±5.05 (4)	**55.93±3.36 (1)**	53.36±4.57 (2)	43.06±2.53 (7)	52.16±3.47 (3)
Ecoli	83.37±4.46 (2)	82.53±5.32 (3)	**83.55±9.4 (1)**	79.02±10.0 (6)	77.7±13.31 (7)	80.65±6.98 (4)	79.39±9.22 (5)
Glass	49.18±11.8 (5)	33.74±9.0 (7)	58.46±9.59 (3)	53.42±16.0 (4)	39.26±13.9 (6)	65.84±9.70 (2)	**67.93±7.65 (1)**
Iosophere	58.4±10.95 (7)	69.23±7.85 (6)	**91.75±4.11 (1)**	90.88±4.0 (3)	79.77±9.6 (5)	91.45±4.24 (2)	88.33±3.87 (4)
Letter	60.42±3.24 (5)	63.28±2.13 (3)	72.3±2.87 (2)	61.61±1.97 (4)	50.36±2.33 (6)	35.47±3.2 (7)	**82.56±1.92 (1)**
German	66.4±3.97 (7)	68.5±3.29 (5)	**71.8±4.21 (1)**	70.0±4.96 (2)	69.1±2.88 (4)	69.8±5.47 (3)	66.6±3.26 (6)
Heart	**84.08±8.77 (1)**	83.7±6.87 (2)	83.33±9.55 (3)	82.96±7.8 (4)	81.11±8.19 (5)	58.89±7.49 (7)	71.11±8.73 (6)
Average Rank	5.2	4.06	**2.4**	3.3	4.9	4.7	3.4

databases except Iris where the two classifiers have the same accuracy.

• A normality test (test of Shapiro-Wilk) done on these databases (Yeast, Ecoli, Glass and Heart) show that they contain attributes that are not normally distributed. We may suppose that applying a Probability-Possibility transformation on the NBC (which leads to NPC) enables the classifier to be less sensitive to normality violation. As suggested in Section 5.3 page 118, one may also think that when normality assumption is not supported by the data, especially for datasets with a high number of attributes, the NBC reinforces the error rate (by the use of multiplication), making the NPC more efficient in this case.

• As previously observed in [107], FNBC is overall better than classical NBC. In fact, FNBC is more accurate than the NBC in seven of the 15 datasets and less accurate in five datasets and not significantly different in three cases (Iris, Diabetes and Satellite Image).

• For the four classifiers using Gaussian distributions (NPC, NBC, FNPC and FNBC), classification results of the FNPC are better than other classifiers for all datasets except in the case of "Transfusion" and "Yeast" databases where FNPC performs worse than others.

• If we compare results for the two flexible classifiers (FNPC and FNBC), we note that the FNPC performs better with the highest accuracy for the majority of datasets. For this classifier, the greatest increase in accuracy compared to the FNBC has occurred for databases "Glass", "Ecoli", "Satellite image", "Segment" and "Letter" (Table 6.1). In Table 4.4, we note that the attributes for these databases range from 8 to 37, and the number of classes from 6 to 26. So the FNPC is significantly more efficient than FNBC (and also than NPC and NBC) for datasets with a high number of attributes and classes.

• Experiments of the second family made of the approximate equality-based classifiers (FuHC, NNPC and NNH) show that they have a competitive efficiency with respect to other possibilistic classifiers for the majority of databases. Besides, we note that the leximin-based NNH, not only outperforms the FuHC and also the NNPC, but also all other classifiers for 5 datasets (Magic, Satellite Image, Segment, Glass and Letter). Table 4.4 shows that these datasets have a higher number of attributes, classes and instances. Thus, the leximin-based NNH seems to be the most efficient classifier for datasets with high dimensionality. Indeed, in contrast with the product-based evaluation, the leximin evaluation is not very sensitive to the dimension of the attributes universe and then the methods based on this evaluation may be expected to be more robust.

The average ranks given between parentheses in Table 6.1 confirm what we have already noted above. On average, the FNPC ranks the first (with rank 2.4) while the FNBC and the NNH ranks the second (respectively 3.3 and 3.4).

For the validation of these results, we compare the behaviour of the possibilistic classifiers by means of a paired t-test. It is a parametric test that checks if the difference between the results of two classifiers over various data sets is significant enough [50]. If the null hypothesis (the two compared classifiers have the same accuracy) is rejected, this means that there are statistically significant differences between the two classifiers. We recall that the p-value measures the importance of this difference. The lower the p-value with respect to a threshold (usually 0.05), the more significant the difference between the classifiers.

Figure 6.1 shows the results of the paired t-test between the FNPC and all the other classifiers, whereas Figure 6.2 shows results between the NNH and all other classifiers. We choose to compare the two best ranked possibilistic classifiers with others for a deeper comparison. In Figure 6.1 (respectively Figure 6.2) and for all

comparisons, dots above the abscissa axes reflect data sets for which the FNPC (respectively the NNH) is significantly better than the compared classifier. Dots under the abscissa axes reflect data sets for which the FNPC (or the NNH) is significantly worse than the second classifier. For the datasets where the two classifiers have an equivalent accuracy ($p - value > 0.05$), dots are on the abscissa axes. The data sets in these comparisons are considered in the same order as in Table 4.4 page 110 (Except that we ignored the data set "Block" for this experiment).

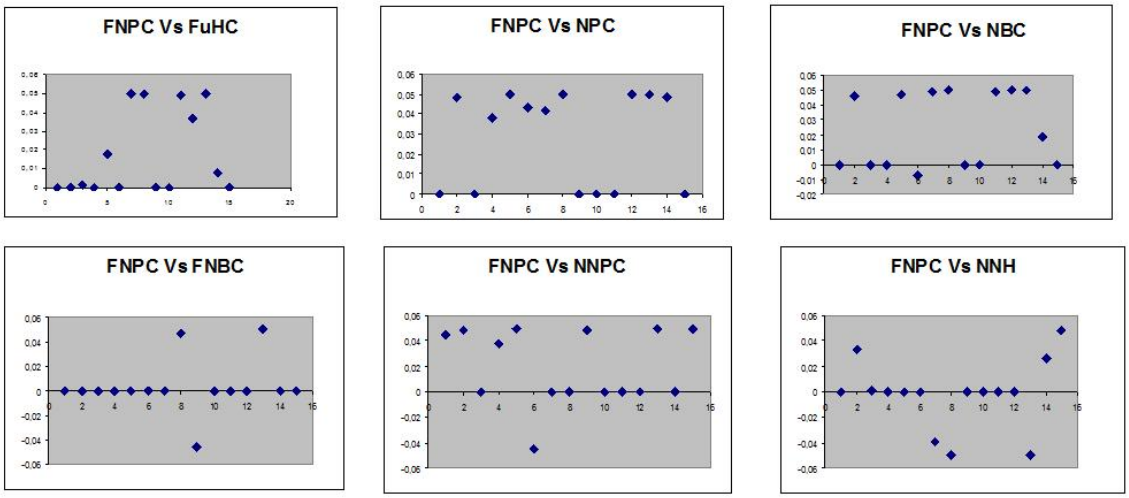

Figure 6.1: Results of the paired t-test between the FNPC and other classifiers

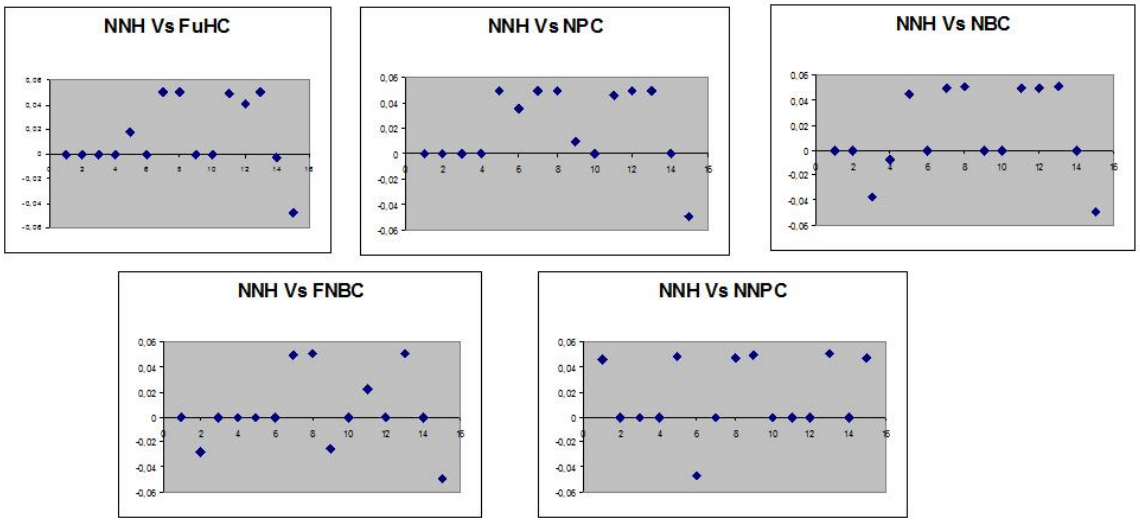

Figure 6.2: Results of the paired t-test between the NNH and other classifiers

Results of the paired t-test shows that the proposed FNPC *significantly outperforms* the FuHC, NPC, NBC, and the NNPC in terms of the number of data sets where the FNPC has a significantly better accuracy than the compared classifier. We can also see in Figure 6.1 that the FNPC is slightly more accurate than the FNBC (because it is significantly more accurate than the latter in 2 data sets and

less accurate in only one data set) and is equivalent in terms of accuracy to the NNH (it is significantly better in 3 data sets, worst in 3 others and equivalent in the remaining data sets).

By comparing the NNH with the other classifiers, we observe similar results as for the FNPC. In fact, the paired t-test in Figure 6.2 proves that the NNH is *much better* than any other classifier except for the FNBC where the NNH is better on 4 datasets, worst on 3 datasets and equivalent on the remaining.

These results show that the FNPC and the NNH are the most efficient possibilistic classifiers among the proposed ones, and they at least compete with the classical and the Flexible Bayesian classifiers. Especially, they are slightly better for datasets with a large number of attributes, classes and instances.

Results of the ambiguity study between near classes

As explained in Section 5.4.3, we are interested in a possible relationship between classification ambiguity and classification errors in the case of possibilistic and Bayesian classifiers.

For each classifier, we fix n levels ($n = 5$ in this experiment) of ambiguity using n intervals having the same length that partition the interval [0, 1]. Then for each ambiguity interval, we compute the number of correctly classified examples (CCE) and the number of incorrectly classified ones (ICE) in the testing set. Experimental results for the NNPC, NPC and the NBC are given respectively in Figure 6.3, Figure 6.4 and Figure 6.5. In each figure, we present the amount of ICE and CCE for each classifier for datasets "Segment" and "Sat-Image" (part *a.* and *c.*). We also exhibit the frequency of error calculated by the ratio: $ICE/(CCE + ICE)$ for the two datasets in part *b* and *d* in each figure. Figure 6.6 summaries results of the error frequency comparison between the three studied classifiers.

We note that ambiguity levels (AL_i in Figures 4, 5 and 6) represent the n intervals of the possibility/probability difference between the most relevant classes ranging in [0, 1] and they are chosen in a decreasing manner such that AL_1 corresponds to the highest ambiguity level whereas AL_n corresponds to the lowest ambiguity level. Results given in Figures 6.3, 6.4 and 6.5 for the CCE and the ICE are an averaged number though the 10-cross-validations for the NNPC, NPC and NBC.

In Figures 6.3 and 6.4, we can see that the frequency of incorrectly classified instances (part b. and d.) decreases when the ambiguity decreases. These figures illustrate also that the highest frequency of incorrect classified instances corresponds to the case of the first ambiguity level that reflects the highest ambiguity. We also notice that, for the lowest ambiguity level (AL_4 and AL_5), possibilistic classifiers

make almost no error ($ICE \approx 0$ even if CCE is always relatively high). These results are nearly the same for the two classifiers NNPC and NPC for almost all datasets. Here we keep only the "Segment" and "Sat-Image" as an illustrative example.

From Figures 6.3 and 6.4 we can see that the higher the ambiguity, the greater the error rate is and the lower the ambiguity is, the more the classifier is able to detect the correct class. So we can say that there is a relationship between ambiguity and classification accuracy for possibilistic classifiers. These results are clearly confirmed by the results shown in Figure 6.6.

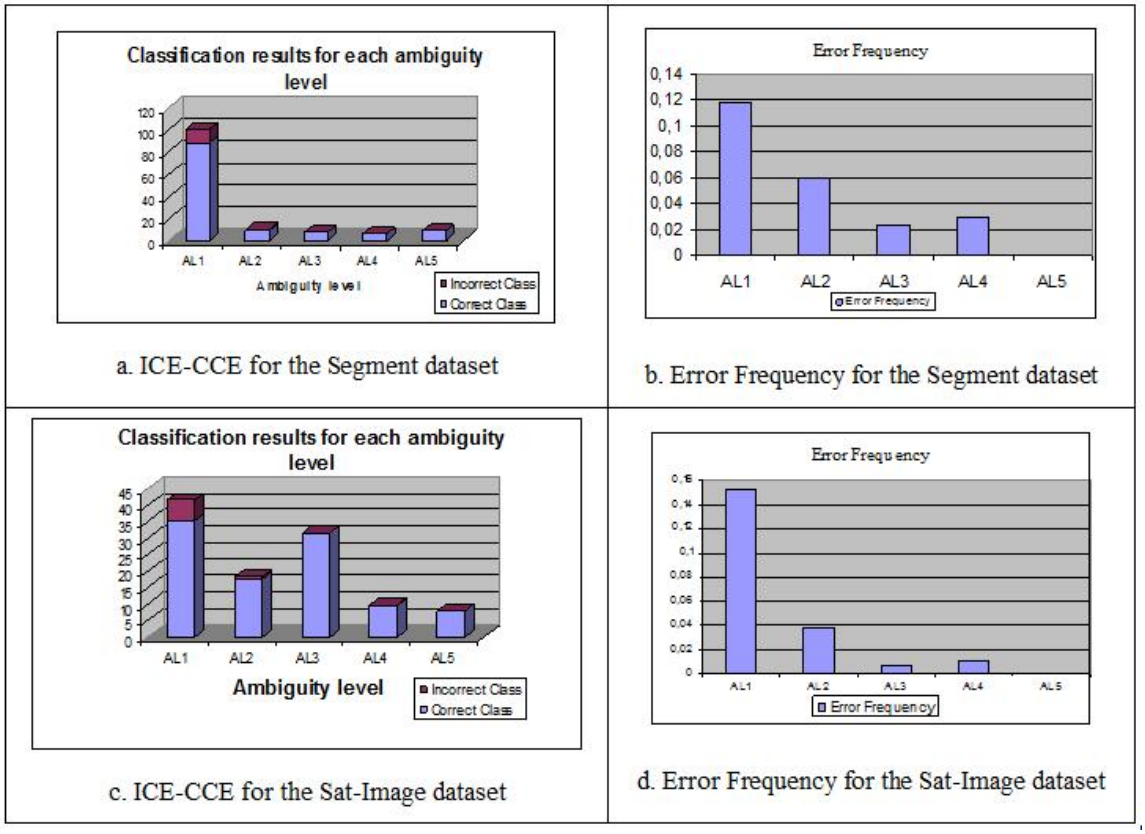

Figure 6.3: Results for the NNPC

However in Figure 6.5 (and also in Figure 6.6) corresponding to the case of NBC, we note that the frequency of error has a non steady behavior. For the two data sets "Segment" and "Sat.Image", instances are either classified with a high ambiguity in AL_1, or much discriminated in AL_5. Moreover, the error rate for this classifier seems to be greater for the lowest ambiguity level than that for the highest one. The error frequency remains higher than 30% for the lowest ambiguity level. So, we can say that in spite of the fact that the NBC distinguishes well between classes in AL_5, it makes more errors in classification. This means that the high distinction ability between classes in this case has no particular meaning, and may be simply caused by the exponential nature of Gaussian densities.

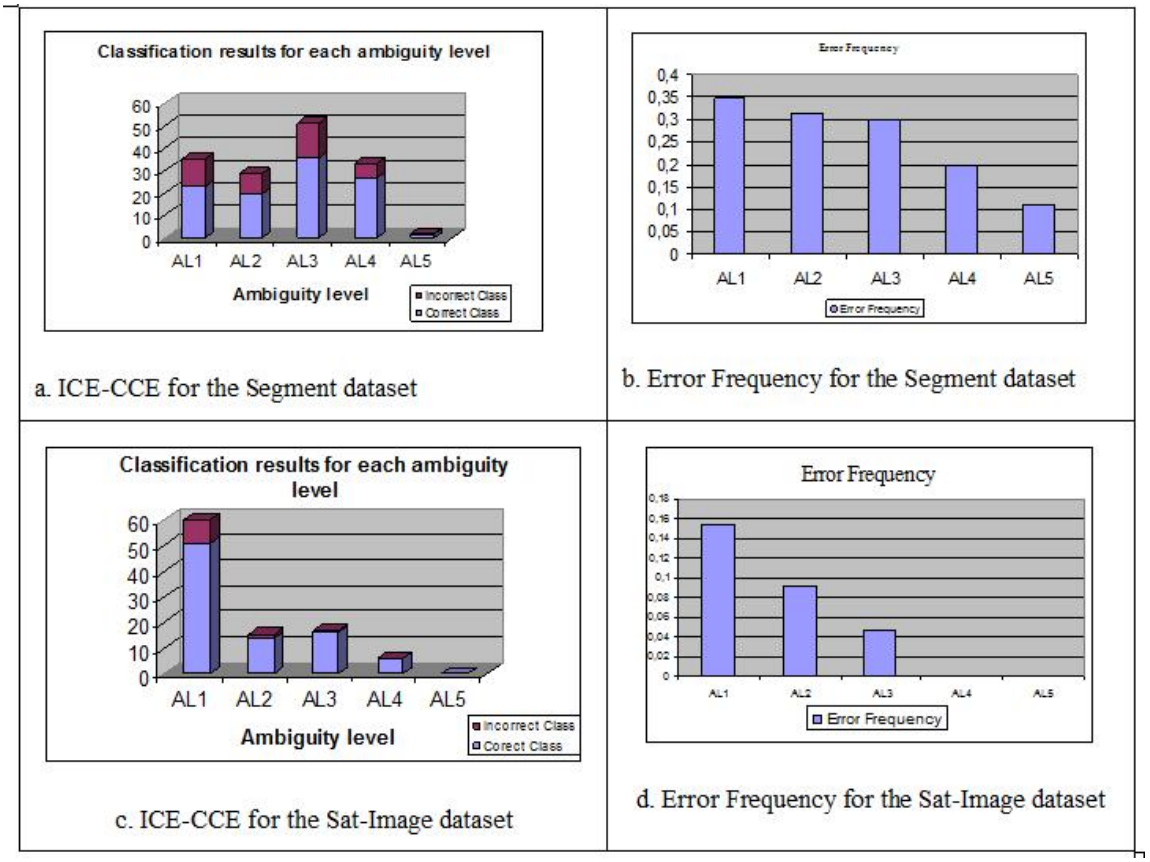

Figure 6.4: Results for the NPC

These results support the intuition underlying the use of possibilistic classifiers. In fact, this study shows that these classifiers are able to detect conflicts in case of ambiguous classification, and to acknowledge difficulties in classifying a conflicting instance. On the contrary, Bayesian Classifiers, due to the difficulty to have a faithful and general measure of ambiguity, seem to have a lower capability for detecting such conflicting situations.

Results of the Hybrid Possibilistic Classification

Table 6.2 includes experimental results for NPC, NBC, FuHC and NNPC in the case of Hybrid Possibilistic Classification.

In this case, we use the Nearest Neighbor Heuristic to help classifying a new instance (instead of only considering the main classifier), when classes have very close plausibility evaluations, i.e., if the difference between their plausibility is less than a fixed level. In our experimental study, this level is fixed to 0.1, (i.e. ambiguity level greater than 0.9), for all classifiers. We choose a relatively high threshold in order to show the effect of the hybrid classification for all possibilistic classifiers at the same time. In fact, the FNPC distinguishes well between classes when compared

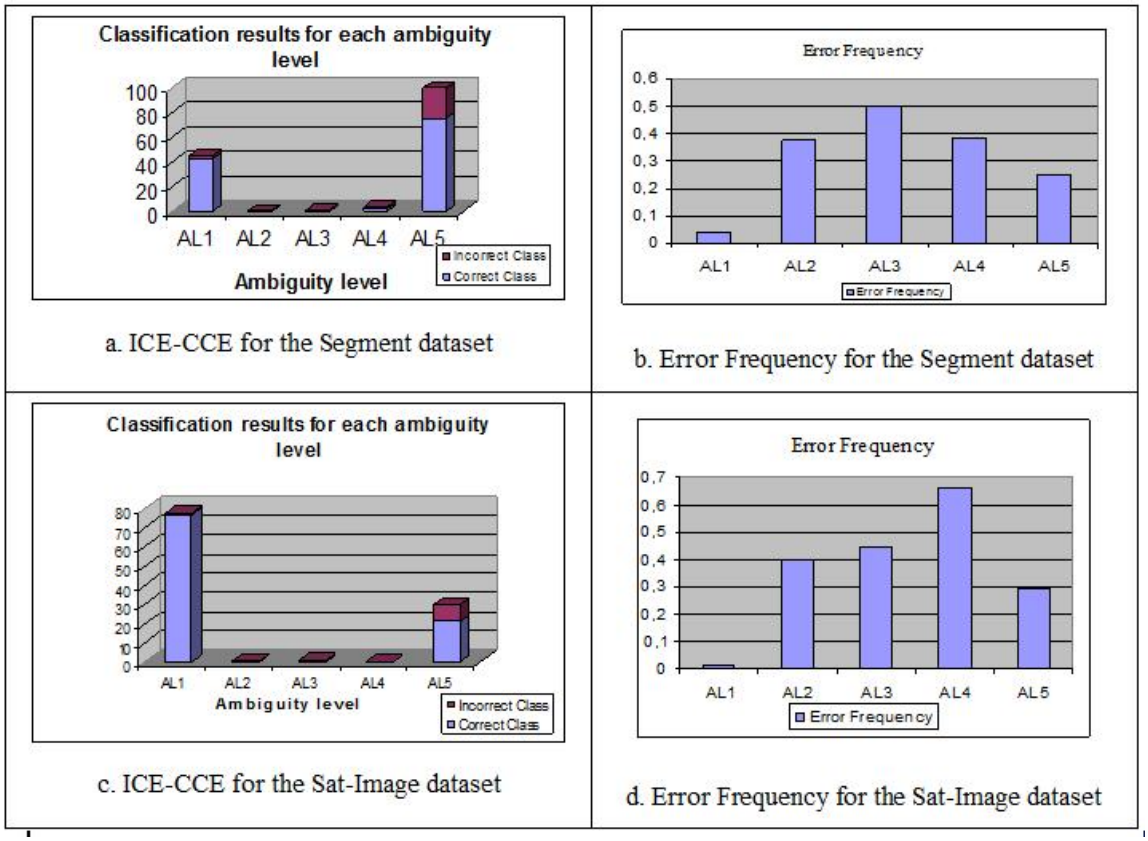

Figure 6.5: Results for the NBC

to NPC or FuHC (the difference between class possibilities is relatively high) so with a low threshold, the hybrid classification would not have any effect on the classical FNPC.

We evaluate the effect of the hybrid classification and its ability to improve the accuracy of possibilistic classifiers. For each classifier, we compare the classification accuracy with or without applying the NNH. For example, in the case of the NPC, we compare column 2 in Table 6.1 with column 2 in Table 6.2.

We are only interested here in knowing if the hybrid classification method improves the initial classifier. For doing this, we use the Wilcoxon Matched-Pairs Signed-Ranks Test as proposed by Demsar [50], since it allows for a direct comparison of the methods without resorting to an analysis of the results on each data set (as done with the paired t-test). It is a non-parametric alternative to the paired t-test that enables us to compare two classifiers (or two versions of the same classifier) over multiple datasets. The Signed-Ranks Test ranks the differences in accuracy for each dataset without regard to the sign of the difference and compares the ranks for the positive and the negative differences.

Table 6.3 includes the p-values for the comparison of each classical possibilistic

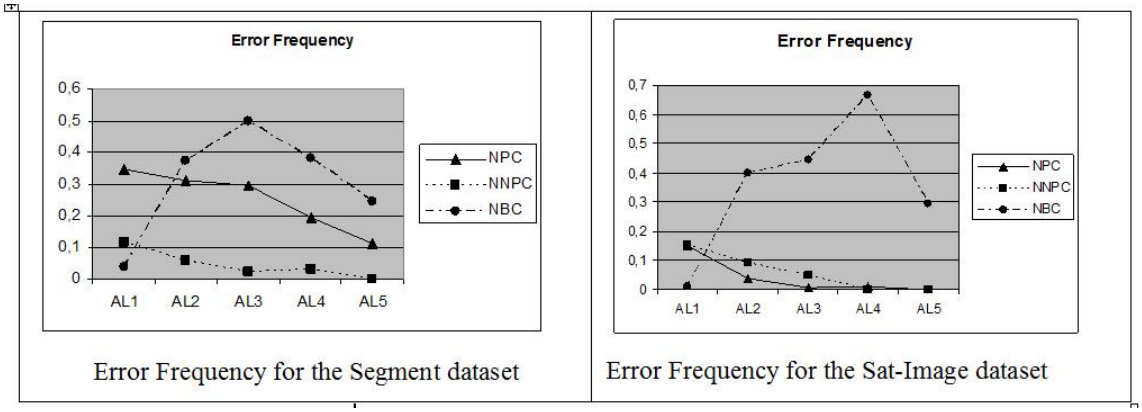

Figure 6.6: Error frequency for the three classifiers

classifier with its hybrid version where we combine this classifier with the NNH.

Results in Table 6.3 show that the hybrid classification contributes to significantly improve the accuracy of the NPC, the FuHC and the NNPC ($p - value < 0.05$). Although there is an improvement of accuracy in the case of the FNPC for some data sets ("Transfusion", "Segment", "Yeast", "Glass", and "Letter"), this improvement is not statistically significant for all data sets ($p - value \leq 0.1016$). By comparing the accuracy of the hybrid version of FNPC with the classical FNBC over the 15 data sets, we note that the FNPC + NNH is better than the FNBC with a $p - value \leq 0.00488$ (instead of a $p - value \leq 0.05225$ when comparing classical FNPC with FNBC).

These results are not surprising since we have already seen in the first experimental study that the NNH is better than the NPC, FuHC and the NNPC and it is equivalent in terms of accuracy to the FNPC. So we can conclude that combining the NNH with possibilistic classifiers in the hybrid approach contributes only to significantly improve the accuracy of classifiers with lower performance than that of the NNH. However, the hybrid classification does not contribute to significantly improve the performance of the FNPC because the NNH and the FNPC have almost the same classification performance.

6.2.2 Results for the extended version of Possibilistic classifiers

As shown in the preliminary study, the probability to possibility transformation based classifiers (NPC and FNPC) are promising and seem to be more efficient, at the same time, than proximity based classifiers and Bayesian classifiers. This is give us the intuition to more investigate this family of classifiers. For this purpose, we first reevaluate these classifiers and Bayesian ones by taking into account prior possibility (or probability) distribution over classes. Even for the perfect case in this

Table 6.2: Experimental results for the Hybrid Possibilistic Classification

	NPC+NNH	FNPC+NNH	FuHC+NNH	NNPC+NNH	NNH
Iris	96.67±4.47	96.67±6.15	96.66±3.34	96.0±6.11	96.0±4.42
Cancer	95.18±1.83	97.36±2.85	96.35±2.27	95.76±3.1	96.06±1.82
Wine	94.93±4.63	97.22±3.73	93.19±4.32	93.89±3.89	93.26±5.98
Diabetes	71.49±4.66	74.1±5.42	69.03±4.29	68.21±5.32	67.97±5.73
Magic	65.46±6.73	74.95±3.23	73.37±4.92	72.72±3.13	74.21±4.51
Transfusion	65.78±6.11	72.22±5.81	71.02±4.35	72.33±2.97	68.73±5.61
Sat. Image	88.53±4.94	92.57±2.48	92.48±1.35	95.05±1.55	93.95±2.6
Segment	75.67±3.02	92.93±2.31	91.73±1.91	95.6±2.09	95.07±1.61
Yeast	54.78±2.83	54.99±3.34	54.51±3.24	48.78±2.02	52.16±3.47
Ecoli	84.26±5.5	84.47±5.54	81.14±8.71	80.47±6.01	79.39±9.22
Glass	59.66±9.75	68.42±9.68	50.0±10.79	66.34±5.42	67.93±7.65
Iosophere	62.71±6.22	92.3±3.15	86.6±7.13	88.34±5.55	88.33±3.87
Letter	68.29±3.14	76.95±2.42	67.1±5.41	50.79±2.96	82.56±1.92
German	69.20±3.12	68.7±3.41	68.3±3.66	67.4±4.39	66.60±3.26
Heart	82.96±7.98	81.85±6.3	78.15±8.19	71.85±6.02	71.11±8.73

Table 6.3: Results for the Wilcoxon Matched-Pairs Signed-Ranks Test

NPC Versus (NPC + NNH)	**FNPC Versus (FNPC + NNH)**	**FuHC Versus (FuHC + NNH)**	**NNPC Versus (NNPC + NNH)**
$p \leq 0.002441$	$p \leq 0.1016$	$p \leq 0.03271$	$p \leq 0.04187$

experiment, we suppose here that prior possibility $\pi(c_j)$ is no longer equal to 1 in equation 5.2 (as in the previous Section) but is rather obtained by computing the probability-possibility transformation (using equation 1.22) of the prior probability distribution over the classes. In the second time, we also experiment the extended version of possibilistic classifiers namely the NPC-2 and FNPC-2 based on computing possibility distribution as a family of Gaussian distribution.

Table 6.4 shows the classification performance obtained with the NPC, NPC-2, NBC, FNPC, FNPC-2 and FNBC for the fifteen mentioned data sets (for the remaining sections, we choose to experiment the "Block" data set instead of the "Letter").

If we compare results in Table 6.4 when prior possibility of classes has been considered for computing final distributions with initial results previously given in Table 6.1, we can see that:

• The performance of possibilistic (NPC and FNPC) and Bayesian classifiers has

Table 6.4: Experimental results for the extended version of Possibilistic Classifiers given as the mean and the standard deviation of 10 cross-validations

	NPC	**NPC-2**	**NBC**	**FNPC**	**FNPC-2**	**FNBC**
Iris	94.66 ±4.99	95.33±4.99	95.33±4.27	**96.0±4.42**	**96.0±4.42**	95.33±6.0
Cancer	95.46±2.02	95.46±2.02	96.19±0.97	**97.66±0.72**	**97.66±0.72**	**97.66±0.72**
Wine	95.48±4.18	95.48±4.18	97.15±2.86	**97.78±2.72**	**97.78±2.72**	96.6±3.73
Diabetes	72.13±5.8	72.91±5.51	75.52±2.67	75.91±3.24	**76.17±3.58**	75.64±3.56
Magic	59.51±6.56	59.32±6.33	65.93±2.91	**73.19±2.63**	73.0±2.49	72.26±2.39
Transfusion	74.63±6.33	74.36±6.32	75.02±5.56	76.63±5.63	**76.76±5.73**	75.7±6.19
Sat. Image	90.46±3.98	90.46±3.98	90.83±3.58	**92.02±1.79**	91.28±3.16	90.55±3.15
Segment	74.13±3.26	74.46±3.44	80.87±2.37	91.0±2.48	**91.13±2.73**	88.6±3.48
Yeast	56.87±3.41	57.68±3.36	46.97±4.69	**58.42±2.27**	58.36±2.14	52.9±3.73
Ecoli	83.7±4.8	83.08±5.47	81.27±5.16	85.51±5.38	**85.8±5.6**	75.82±7.1
Glass	47.27±14.54	46.32±14.79	43.12±8.12	**67.9±10.77**	67.38±9.86	57.97±8.98
Iosophere	60.38±9.99	60.95±9.1	70.09±6.15	91.75±4.3	**92.62±5.05**	92.05±5.0
Block	88.29±1.4	88.49±1.86	89.66±3.22	93.33±0.98	**93.51±1.07**	90.21±2.14
German	73.1±2.98	73.2±2.99	73.0±2.97	75.6±3.07	**75.7±2.93**	70.0±4.22
Heart	84.08±4.98	**84.45±4.32**	83.34±5.56	83.34± 5.56	83.7±5.79	84.08±6.15

reported an increase in accuracy for the second version of these classifiers. Especially for the NPC and FNPC, the improvement can be seen in 8 of the 15 data sets and the greatest increase is in the data sets "Transfusion", "Yeast", "Glass" and "German" (For example the accuracy of the "Glass" becomes 67.9 instead of 53.42 for the FNPC). Similar improvement is also noticed in the case of NBC and FNBC in particular for the "Transfusion" and "Glass". For these datasets we note that classes are not equivalently covered by instances, i.e: there is one dominant class highly covered and other classes weakly represented by examples. Thus we can say that, considering prior possibilities of classes $\pi(c_i) \neq 1$ obtained by a probability-possibility transformation of the prior probability distribution, enables to have a more realistic representation of data which contributes to this improvement.

If we compare the classification performance of the six classifiers in Table 6.4 we note that:

• The NPC (the improved version) and the NPC-2 has very close results except for the "Yeast" (respectively "Glass") were the NPC-2 (Resp. NPC) is slightly better than the NPC (resp. NPC-2). Results for the FNPC and FNPC-2 are also equivalent except for the case of "Sat.Imge" and "Iosophere". So the two families of Naive possibilistic classifiers obtained from the two probability-possibility transformation methods has close performances.

• As for the initial version of NPC, the NPC-2 is significantly more accurate than

the NBC in data sets ("Yeast", "Ecoli", "Glass", "German" and "Heart") and less accurate on the remaining databases except for the "Iris" where the two classifiers have the same accuracy.

• These results reinforce the idea previously mentioned when comparing the classical NPC and NBC [23]. In fact, the NBC seems to partially fail when classifying instances in data sets with attributes that strongly violate the normality assumption (a Shapiro-Wilk test done on these data sets prove the normality violation). This can be explained by the fact that the NBC reinforces the error rate (by the use of multiplication) especially for data sets with a high number of attributes

• We expect that, if the normality assumption is strongly confirmed for a given data set, it is better to use a probability distribution for classification since it remains more precise. In the other case, we may suppose that applying a Probability-Possibility transformation on the NBC (which leads to NPC [23] or to NPC-2 [20]) enables the classifier to be less sensitive to normality violation.

• The flexible possibilistic classifiers (FNPC and FNPC-2) significantly outperform the FNBC for the majority of data sets. As observed before, the highest increase in accuracy compared to the FNBC has occurred for databases "Yeast", "Glass", "Ecoli", "Segment", "German" and "Block" (Table 6.4) having hight number of attributes and classes. We can conclude that the FNPC and FNPC-2 are significantly more efficient than the FNBC (and also other classifiers) for data sets with a high dimentionality.

• For the six classifiers, classification results of the two flexible possibilistic classifiers (FNPC and FNPC-2) are largely better than other classifiers for all data sets except in the case of "Iris", "Cancer" and "Heart" databases where the FNPC-2 (and also FNPC) have almost the same accuracy as others.

6.3 Experiments of possibilistic classifiers for the imperfect numerical data

This section provides experimental results of the uncertain versions of possibilistic classifiers [25] [20]. Although uncertainty in databases is a important issue in machine learning, there are no uncertain nor imprecise data sets which could be used for testing algorithms dealing with such type of data. For this reason, we first give here a heuristics to create uncertainty and imprecision in an artificial manner. In the second part of this section we present the criteria suitable for evaluating the classification accuracy of possibilistic classifiers in the imperfect case. Finally, we give results for the imperfect case.

6.3.1 Generation of imperfect data

Data sets described in Table 4.4 are initially perfect with certain and precise attributes and classes. In order to evaluate possibilistic classifiers in the imperfect case, we have artificially introduced imperfection in these data sets by transforming the original precise and certain instances into imperfect ones.

Creating possibilistic labels: Uncertainty on the training set is created by replacing the certain class label of each instance by a possibility distribution over class labels. To generate a possibility distribution, we suppose that we have two independent experts and that they are, to some extent, unable to classify each training instance in a certain manner. So we ask each expert to give a possibility distribution over class labels reflecting his/her knowledge about this uncertain situation. Then we apply an information fusion procedure [71] to produce the final possibility distribution for each instance. Each expert may simply be a possibilistic classifier trained on the perfect (certain and precise) data set. In this experiment we have used the certain FNPC and the FuHC classifiers, as presented in Chapter 5, to simulate experts. For information fusion, we apply a *disjunctive operator* [71], as introduced in Section 1.4.5 page 30, to create the final possibility distribution $\pi_{a_{tr}}$:

$$\forall \omega \in \Omega, \pi_{\vee}(\omega) = \oplus_{i=1..n} \pi_i(\omega) = \max_{i=1}^{n} \pi_i(\omega) \tag{6.1}$$

We prefer the disjunctive operator to the conjunctive one since the two classifiers may disagree and we cannot be sure which one is more reliable. Moreover, possibilistic distributions generated with this operator cover the imprecise case where more than one class may have a possibility degree equal to 1. We create uncertain training set by the following:

1. Train the FNPC and the FuHC using the original crisp training set.
2. Use the obtained possibilistic classifiers to predict the class labels. for each training instance.
3. For each training instance a_{tr}, fuse the two possibility distributions obtained from each classifier using a *disjunctive operator*.
4. Keep the attribute values of each instance in the training set unchanged and replace the crisp class label by $\pi_{a_{tr}}$.

Creating imprecise attributes: Attributes in the testing set are made uncertain in the following way. In each testing instance, we convert each attribute value into

an uncertain interval. For each attribute, we scan all of its value in the database and get its minimum value X_{min} and its maximum value X_{max}. Then we replace each attribute value by a generated interval I = [L, U] in order to create imprecision on this attribute. If x is the perfect value of the current attribute, its lower bound L(Resp. upper bound U) is calculated as follows: $L = x - (x - X_{min}) * rand1$ (resp. $U = x + (X_{max} - x) * rand2$), where $rand1$ and $rand2$ denote two random numbers reflecting the uncertainty level $AttrL$ on this attribute. $AttrL$ is a level which describes the larger of the interval and takes values in $\{0.25, 0.5, 0.75 \text{ or } 1)\}$. For each level $AttrL$, we generate an uncertain dataset U_{AttrL} where $rand1$ and $rand2$ range between 0 and $AttrL$. Hence, for each perfect testing set, we create four uncertain datasets $U_{0.25}, U_{0.5}, U_{0.75}$ and U_1.

6.3.2 Classification accuracy measures

To measure the accuracy of possibilistic classifiers, we use two evaluation criteria:

- **The percentage of Most Plausible Correct Classification (MPcc):** counts the percentage of instances whose all most plausible classes, predicted by the possibilistic classifier, are *exactly the same* as their initial most plausible classes given by the possibility distribution labeling each testing instance.

$$MPcc = \frac{Number\_of\_exactly\_well\_classified\_instances}{Total\_nbr\_classified\_instances} * 100 \tag{6.2}$$

- **The Information Affinity-based Criterion (AffC)** [102][104]: is a degree of affinity between the predicted and the real possibility distribution labeling the testing instances

$$InfoAffC = \frac{\sum_{i=1}^{n} Aff(\pi_i^{real}, \pi_i^{pred})}{Total\_nbr\_classified\_instances} \tag{6.3}$$

$$Aff(\pi_1, \pi_2) = 1 - \frac{d(\pi_1, \pi_2) + Inc(\pi_1, \pi_2)}{2} \tag{6.4}$$

where $d(\pi_1, \pi_2)$ is the Manhattan distance between π_1 and π_2 and $Inc(\pi_1, \pi_2) = Inc(\pi_1 \wedge \pi_2)$ is the degree of inconsistency between π_1 and π_2 which ranges in [0,1] and is calculated as follows:

$$Inc(\pi) = 1 - \max_{\omega \in \Omega} \{\pi(\omega)\}. \tag{6.5}$$

6.3.3 Results for the imperfect numerical data

This experimental study is divided in two parts. First, we evaluate the uncertain possibilistic classifiers to handle uncertainty only in class attribute and we keep attributes in the testing set perfect. Second, we test the accuracy of the proposed classifiers when attributes in the testing set are uncertain whereas training set is kept perfect. We choose to test each uncertainty type independently in order to check the efficiency of possibilistic classifiers to deal with each situation.

1. Uncertainty type 1: Uncertain classes

Table 6.5: Experimental results for uncertain classes given as the mean and the standard deviation of 10 cross-validations

	FNPC		FNPC-2		FuHC		NNPC	
	MPcc	**AffC**	**MPcc**	**AffC**	**MPcc**	**AffC**	**MPcc**	**AffC**
Iris	94.0±3.6	0.94±0.0	94.0±3.6	0.94±0.01	93.33±6.7	0.95±0.0	88.67±7.3	0.88±0.01
Cancer	96.19±1.8	0.98±0.0	96.19±1.8	0.98±0.0	95.31±2.0	0.99±0.01	42.05±13.4	0.79±0.01
Wine	92.64±6.2	0.94±0.01	92.64±6.2	0.94±0.01	92.08±5.2	0.95±0.0	87.08±8.6	0.8±0.01
Diabetes	76.72±6.6	0.96±0.0	76.85±6.4	0.96±0.0	54.41±6.7	0.95±0.0	58.6±5.7	0.95±0.0
Magic	74.4±3.7	0.93±0.0	74.4±3.7	0.93±0.0	61.27±10.0	0.94±0.01	66.2±4.7	0.92±0.01
Transfusion	84.1±4.1	0.98±0.0	83.57±4.2	0.98±0.0	62.46±6.4	0.98±0.0	51.89±5.3	0.98±0.0
SatImage	90.55±3.0	0.98±0.01	90.55±3.1	0.98±0.01	90.37±3.3	0.98±0.01	89.18±3.3	0.7±0.01
Segment	70.87±5.0	0.92±0.01	70.93±5.1	0.92±0.01	63.13±4.4	0.95±0.0	79.6±3.3	0.8±0.01
Yeast	58.08±3.6	0.96±0.0	58.28±3.6	0.96±0.0	20.15±3.8	0.93±0.01	22.1±2.8	0.9±0.0
Ecoli	80.65±8.1	0.93±0.01	80.36±8.2	0.93±0.01	65.1±13.8	0.91±0.01	78.0±6.4	0.88±0.01
Glass	53.93±13.6	0.92±0.02	52.54±13.3	0.92±0.02	35.99±13.0	0.9±0.08	44.46±14.6	0.83±0.02
Iosophere	78.34±6.9	0.94±0.02	78.63±6.9	0.94±0.02	75.2±9.1	0.93±0.02	76.34±7.9	0.85±0.02
Block	78.29±2.3	0.93±0.01	78.69±1.07	0.94±0.0	75.59±2.2	0.87±0.01	74.97±2.1	0.64±0.0
German	81.3±4.0	0.96±0.01	81.2±3.9	0.96±0.01	77.0±3.7	0.95±0.01	17.7±3.4	0.88±0.01
Heart	89.26±6.7	0.96±0.01	89.26±6.5	0.96±0.01	89.26±8.2	0.97±0.01	48.89±5.4	0.82±0.02

Table 6.5 shows the classification performance (MPcc and InfoAffC criterion) obtained with the FNPC, FNPC-2, FuHC and NNPC for the fifteen uncertain data sets [25] [20].

If we analyze results in Table 6.5, we note that:

• For the uncertain FNPC and FNPC-2, 5 of the 15 data sets have reported an increase in accuracy if compared to the perfect case, 10 of the 15 data sets have reported a decrease in accuracy (see Table 6.4 for comparison) but in 6 of the 10 the decrease is less than 5% and the highest decrease is for the "Segment" which is about 20% but the MPcc remains $> 70\%$.

• For the proximity based classifiers the decrease in accuracy is more considerable and the highest one is reported for the "Yeast" in the case of FuHC and "Cancer" and "German" in the case of NNPC which is respectively about 30% (FuHC) and 50% (NNPC).

• From this table we note that the two flexible possibilistic classifiers have close accuracy as in the perfect case. To compare their results to the two other proximity based classifiers in terms of MPcc, we use the Wilcoxon Matched-Pairs Signed-Ranks Test as proposed by Demsar [50]. Comparison results given in Table 6.6 show that the FNPC (and also FNPC-2) is always significantly better ($p - value < 0.05$) than the two proximity based classifiers for all data sets whereas the FuHC and NNPC have competitive performance.

• As reported in the perfect case from these results we can say that, overall the FNPC and FNPC-2 shows a high ability to detect the most plausible classes even for uncertain data sets with high level of uncertainty (all training instances are assumed to be uncertain). In this study, we have used a rigid MPcc criteria which considers an instance as incorrectly classified if the difference between predicted and real full plausible classes is at least equal to 1. Although this rigid criteria and even for 100% uncertainty level, the two flexible possibilistic classifiers show a high ability to deal with imperfect instances almost as good as with perfect ones (See Table 6.4).

• By analyzing the InfoAffC criteria we can see that the values are high for the different classifiers and for almost all data sets. For all data sets, the InfoAffC is > 0.9 for the FNPC, FNPC-2 and FuHc (except in the case of Block which is close to 0.9) and > 0.7 for the NNPC. From these results, we can conclude that the possibilistic classifiers are able to predict possibility distributions *highly* consistent with the initial uncertain distributions.

• For the majority of data sets, the InfoAffC criteria confirms the results reported by the MPcc. However we can see a significant divergence between the values of InfoAffC and MPcc for some data sets and mainly for the NNPC (for example, for the "Yeast" the MPcc value is 22.1% and the AffC is 0.9). If compared to the perfect case, the MPcc decrease is about 20% for the Yeast and 50% for the Cancer. Also for the two flexible classifiers and for some data sets ("Segment", "Glass", "Iosophere" and "Block") there is a significant decrease in MPcc, if compared to the perfect case which is respectively about 20 %, 15%, 13% and 15% however the InfoAffC remains higher than 0.9.

• This divergence means that for many testing instances, the possibilistic classifier provides possibility degrees *too close* to the initial possibility distribution (high InfoAffC) but the predicted and real full plausible classes are *not exactly* the same (weak MPcc). So we can say that this decrease in accuracy for these data sets returns to the rigid nature of the MPcc criteria which causes the absence of classification

for many instances in the data set where the classifier provides more than one fully plausible class which are not exactly the same as those given in the real distribution. This is mainly the case for data sets with large number of classes.

• For the NNPC the decrease is more significantly observed since this latter, by using the maximum operator in formula 5.12 page 131, looks only for *one nearest neighbor* which makes conditionals possibilities on classes to tend to 1 for more than one particular class. That's why the NNPC confuses much between near classes and causes a 0 classification percentage for many instances in the data set where the classifier provides more than one fully plausible class whereas in the real distribution the number of fully plausible classes is fewer.

• The results of the NNPC could be improved if we consider a more relaxed MPcc criterion for which we allow to an instance to be classified with a particular percentage $p \in [0, 1]$, for example $p = 1/2$ if only one full plausible class is in the initial distribution among two full plausible classes detected by the classifier. By applying this relaxed criterion, the MPcc for the "Cancer" in the case of NNPC becomes 69.34% (instead of 42.05%).

Table 6.6: Results for the Wilcoxon Matched-Pairs Signed-Ranks Test

FNPC Versus FuHC	**FNPC Versus NNPC**	**FuHC Versus NNPC**
$p \leq 0,005859$	$p \leq 0.01855$	$p \leq 0.8311$

2. Uncertainty type 2: Imprecise attributes

Table 6.7 shows the MPcc and the InfoAffC results obtained with the four classifiers for each imprecision level on attributes and for the fifteen mentioned data sets. C1, C2, C3 and C4 in Table 6.7 are respectively the FNPC, the FNPC-2, the FuHC and the NNPC. By comparing the classification performance we see that the accuracies of the four algorithms decrease when the imprecision level of attributes increases (when intervals are broader). Despite this decrease we note that [25] [20]:

• The FNPC, FNPC-2 and FuHC have reported relatively high performance if compared to the perfect case. We can also note that the decrease in accuracy for the FNPC and FNPC-2 is relatively stable and not acute.

• Despite the decrease in accuracy, we note that the ratio remains high in average mainly for the three first classifiers. For instance, if we analyze the results relative to the FNPC and FNPC-2, we remark that the MPcc remains higher than 60% for the highest uncertainty level (U_1)(the worst case) and this for all data sets except the "Yeast" and "Glass" where the value is respectively about 32% and 43%.

Table 6.7: Experimental results for uncertain attributes given as the mean and the standard deviation of 10 cross-validations

		$U_{0.25}$		$U_{0.5}$		$U_{0.75}$		U_1	
		MPcc	**AffC**	**MPcc**	**AffC**	**MPcc**	**AffC**	**MPcc**	**AffC**
Iris	C1	93.33±8.9	0.95±0.06	90.67±10.0	0.94±0.06	88.0±11.9	0.92±0.06	87.33±10.1	0.9±0.04
	C2	93.33±8.9	0.95±0.05	90.67±10.0	0.94±0.06	88.0±11.9	0.92±0.05	86.67±9.4	0.9±0.04
	C3	92.0±8.8	0.87±0.03	86.67±9.9	0.85±0.04	79.33±17.0	0.83±0.05	66.0±17.8	0.77±0.08
	C4	88.0±5.8	0.77±0.01	82.0±10.3	0.77±0.01	77.33±11.6	0.76±0.01	62.67±10.4	0.73±0.03
Cancer	C1	97.66±1.5	0.98±0.02	96.04±2.5	0.96±0.02	95.9±2.3	0.96±0.02	94.14±2.6	0.94±0.02
	C2	97.66±1.5	0.98±0.02	96.04±2.5	0.96±0.02	95.9±2.3	0.96±0.02	94.28±2.6	0.94±0.02
	C3	96.05±2.1	0.97±0.02	95.9±1.8	0.97±0.01	94.73±3.3	0.95±0.02	92.84±3.6	0.93±0.02
	C4	31.19±4.7	0.78±0.01	29.43±5.8	0.77±0.0	25.63±5.1	0.76±0.0	23.14±5.4	0.76±0.0
Wine	C1	96.6±3.7	0.98±0.02	94.93±3.9	0.96±0.02	91.6±4.4	0.94±0.02	87.08±5.0	0.91±0.04
	C2	95.48±3.4	0.97±0.02	94.93±3.9	0.96±0.03	91.6±3.7	0.94±0.02	85.9±5.3	0.9±0.04
	C3	93.05±8.0	0.9±0.02	90.9±6.9	0.88±0.02	86.18±11.0	0.84±0.05	71.74±24.0	0.77±0.11
	C4	91.67±6.2	0.74±0.01	75.42±9.2	0.72±0.01	62.43±7.9	0.7±0.01	47.71±9.7	0.68±0.03
Diabetes	C1	71.99±3.5	0.77±0.02	68.73±4.9	0.75±0.03	68.09±5.1	0.74±0.04	65.48±4.0	0.72±0.02
	C2	73.17±3.6	0.77±0.02	69.52±4.5	0.75±0.03	68.35±4.8	0.74±0.04	66.0±4.2	0.72±0.02
	C3	72.39±5.6	0.76±0.01	71.35±5.4	0.76±0.01	66.92±6.0	0.75±0.01	59.13±5.6	0.73±0.01
	C4	39.47±6.4	0.75±0.0	39.88±9.1	0.75±0.0	37.68±8.9	0.75±0.0	35.86±8.1	0.75±0.0
Magic	C1	71.41±5.0	0.77±0.03	72.34±3.3	0.78±0.02	75.32±5.0	0.79±0.02	70.86±3.9	0.76±0.02
	C2	73.29±4.2	0.78±0.02	72.46±3.2	0.78±0.02	72.26±3.1	0.77±0.02	70.77±3.9	0.76±0.02
	C3	67.14±6.2	0.75±0.02	66.3±5.7	0.74±0.02	65.2±7.5	0.73±0.03	60.91±6.2	0.71±0.02
	C4	64.81±7.2	0.75±0.0	60.81±7.7	0.75±0.0	59.69±7.8	0.75±0.0	58.21±7.2	0.75±0.0
Transfusion	C1	65.14±7.8	0.72±0.05	64.22±7.1	0.72±0.04	62.36±7.9	0.72±0.03	61.12±5.2	0.72±0.03
	C2	65.41±7.9	0.73±0.04	63.82±7.1	0.73±0.04	63.02±7.5	0.72±0.03	60.86±5.4	0.72±0.03
	C3	61.94±7.5	0.75±0.01	57.78±7.2	0.74±0.01	53.49±5.1	0.74±0.01	46.11±5.9	0.73±0.01
	C4	6.86±5.5	0.75±0.0	9.26±5.1	0.75±0.0	6.72±5.1	0.75±0.0	6.32±5.8	0.75±0.0
Sat.Image	C1	89.45±2.1	0.93±0.02	86.42±2.9	0.91±0.02	85.96±2.8	0.91±0.02	84.5±3.3	0.9±0.02
	C2	89.27±2.1	0.93±0.01	86.15±3.2	0.91±0.02	85.78±2.6	0.91±0.02	84.4±3.5	0.9±0.02
	C3	87.25±2.9	0.93±0.02	83.21±5.4	0.9±0.03	75.78±8.6	0.85±0.06	60.28±10.0	0.75±0.07
	C4	91.1±2.0	0.68±0.0	87.25±3.1	0.67±0.01	80.55±2.4	0.66±0.01	75.23±3.1	0.66±0.0
Segment	C1	87.8±2.3	0.93±0.01	83.07±3.5	0.91±0.02	77.67±3.4	0.89±0.02	73.6±4.0	0.86±0.02
	C2	87.93±2.3	0.93±0.01	83.13±3.6	0.91±0.02	77.8±3.2	0.89±0.02	73.4±3.6	0.86±0.02
	C3	77.33±2.6	0.88±0.01	73.0±3.9	0.86±0.01	65.4±5.0	0.83±0.02	51.2±2.9	0.75±0.02
	C4	79.67±2.8	0.72±0.01	68.67±4.2	0.7±0.0	59.73±4.2	0.69±0.01	38.2±2.8	0.64±0.0
Yeast	C1	53.37±2.7	0.79±0.02	49.53±4.6	0.77±0.01	40.44±3.1	0.74±0.02	30.73±3.0	0.7±0.01
	C2	53.715±2.8	0.795±0.02	49.335±4.7	0.775±0.01	40.515±3.1	0.745±0.02	31.25±3.0	0.75±0.01

	C3	49.94±3.3	0.69±0.01	41.23±4.3	0.68±0.01	37.93±3.8	0.67±0.01	35.98±5.4	0.67±0.01
	C4	8.49±2.5	0.63±0.0	6.74±1.7	0.62±0.01	4.92±1.4	0.62±0.0	3.78±1.6	0.62±0.0
Ecoli	C1	81.86±5.2	0.91±0.03	77.74±9.2	0.9±0.04	70.28±10.8	0.86±0.04	63.06±6.7	0.83±0.02
	C2	81.27±5.3	0.91±0.03	77.19±9.9	0.9±0.04	70.28±10.5	0.86±0.03	63.65±6.5	0.83±0.02
	C3	77.45±8.2	0.82±0.02	69.14±7.3	0.79±0.01	61.0±7.9	0.77±0.01	53.82±7.2	0.75±0.02
	C4	71.63±6.8	0.73±0.01	64.78±8.1	0.72±0.01	61.68±10.5	0.72±0.01	55.98±9.6	0.71±0.01
Glass	C1	51.075±13.2	0.795±0.04	48.285±13.7	0.775±0.04	45.425±17.5	0.755±0.07	42.555±14.4	0.745±0.06
	C2	49.65±13.0	0.78±0.04	45.53±15.4	0.77±0.05	42.15±18.7	0.75±0.07	41.08±14.7	0.74±0.06
	C3	34.2±9.5	0.71±0.02	31.84±13.2	0.7±0.02	29.96±10.6	0.68±0.02	27.12±13.7	0.67±0.03
	C43	52.81±11.6	0.64±0.02	50.43±7.5	0.64±0.01	45.26±9.0	0.64±0.02	47.62±13.5	0.63±0.02
Iosophere	C1	92.3±2.2	0.92±0.02	90.59±3.2	0.91±0.03	90.31±4.3	0.91±0.04	86.9±5.6	0.88±0.05
	C2	91.16±2.0	0.92±0.02	90.3±3.4	0.91±0.03	89.74±4.3	0.91±0.04	86.9±5.4	0.88±0.04
	C3	79.2±6.4	0.81±0.06	77.77±6.9	0.8±0.06	75.79±7.0	0.77±0.06	74.93±7.4	0.76±0.07
	C4	89.19±6.3	0.79±0.02	87.17±7.2	0.79±0.02	78.62±16.6	0.78±0.02	76.33±16.1	0.78±0.02
Block	C1	89.53±0.7	0.93±0.01	87.1±1.4	0.91±0.01	82.77±1.9	0.88±0.01	75.72±2.4	0.84±0.01
	C2	89.2±1.7	0.93±0.01	86.86±2.2	0.91±0.01	81.2±2.0	0.87±0.01	75.9±3.0	0.84±0.01
	C3	88.42±2.0	0.79±0.01	88.36±2.0	0.78±0.01	88.96±2.4	0.78±0.01	88.58±2.4	0.78±0.01
	C4	89.79±1.9	0.62±0.01	89.24±2.5	0.62±0.0	88.31±3.6	0.62±0.0	87.96±3.5	0.62±0.0
German	C1	71.8±2.79	0.76±0.02	71.0±3.82	0.76±0.0	70.7±3.	0.75±0.02	67.7±4.65	0.74±0.02
	C2	71.5±4.2	0.76±0.02	71.8±4.3	0.76±0.02	69.4±3.2	0.75±0.02	69.2±3.4	0.74±0.03
	C3	69.0±4.1	0.75±0.03	69.4±3.5	0.75±0.02	69.5±4.2	0.75±0.03	69.3±5.0	0.75±0.03
	C4	6.3±4.9	0.75±0.0	5.9±5.8	0.75±0.0	6.2±5.3	0.75±0.0	17.2±7.3	0.75±0.0
Heart	C1	84.08±4.7	0.85±0.04	82.59±5.3	0.84±0.04	82.22±5.9	0.84±0.05	81.85±6.9	0.84±0.05
	C2	84.08±4.7	0.85±0.04	81.85±5.8	0.84±0.04	81.85±6.5	0.84±0.05	81.48±7.0	0.84±0.05
	C3	81.4811±3.9	0.84±0.03	81.11±4.7	0.84±0.03	80.0±6.7	0.83±0.03	79.63±4.5	0.83±0.04
	C4	53.7±11.1	0.75±0.0	44.07±9.1	0.75±0.0	40.0±12.5	0.74±0.01	37.41±7.8	0.74±0.01

The low results reported for the these data sets are not related to the classifier since the MPcc reported for the original certain version of these data sets is about 58% for the "Yeast" and 67% for the "Glass" for the certain FNPC.

• However the NNPC seems to find difficulties when classifying instances with imprecise attributes especially for data sets "Cancer", "Transfusion", "Diabetes", "Yeast" and "German". As reported in the uncertain case (Table 6.5), low accuracies are related to the *max* operator, used in NNPC, combined with the rigid MPcc criterion.

• As in the uncertain case (Table 6.5), the accuracy of the FNPC (and also the FNPC-2) is always (even slightly) better than other classifiers for all uncertainty levels and all databases expect the case of "Glass", in which this classifier performs worse than the NNPC.

• The values of the InfoAffC criterion reported for the different classifiers and for the different data sets are relatively high. For 11 of the 15 data sets, this value remains higher than 0.7, for all uncertainty levels and for the four classifiers and it is higher than 0.6 for the remaining data sets. So, we can say that the predicted and initial possibility distributions are relatively consistent.

From results given in Tables 6.5 and 6.7, we conclude that FNPC and the FNPC-2 are more accurate that the two others and can be considered as good classifiers which are well suitable to deal with perfect or imperfect continuous data and all types of databases. However results of the proximity based classifiers could be improved if i) we use a more appropriate MPcc criterion and ii) we refine theses approaches by using a Nearest-Neighbor heuristic to separate indistinguishable classes [24].

6.4 Conclusion and discussion

Experimental results show the performance of possibilistic classifiers for handling numerical input data.

While proximity based classifiers shows competitive efficiency compared to probability based possibilistic classifiers, they seem to confuse much between near classes (especially the NNPC). Besides the NNH, viewed as an independent classifier, is efficient in particular for classifying databases with high dimensionality. The hybrid classification method exhibits a clear ability to improve the accuracy of possibilistic classifiers, in particular those having a great confusion level between classes which produce close plausibility estimates for classes, such as the NPC.

On the other hand considering priori distribution over classes in the case of extended versions of Gaussian-based Possibilistic classifiers (NPC, NPC-2, FNPC and FNPC-2), contributes to significantly improve their accuracy mainly for data sets with classes not equivalently covered. While Naive possibilistic classifiers (NPC and NPC-2) are less sensible than NBC to normality violation, Flexible possibilistic classifiers (FNPC and FNPC-2) shows high classification accuracy and good ability to deal with any type of data when compared to other classifiers in the same family.

To test possibilistic classifiers in the uncertain case, we have artificially introduced imperfection in data sets from the UCI machine learning repository. Experimental results show the performance of these classifiers to deal with imperfect as well as perfect numerical data. Indeed, the FNPC and FNPC-2 show a high ability to detect the full plausible class labels with possibility distributions very consistent with initial distributions. Possibilistic classifiers exploiting proximity are competitive with others, besides the NNPC has some difficulties to distinguish between near classes, which decreases its performance although predicted possibilities distributions are valuable.

Conclusion

In the first part of this thesis, we have investigated the study of rule induction algorithms and their efficiency in the classification task. The analysis of rule based classifiers let us to show that they suffer from major drawbacks when classifying test examples. Especially, the multiple classification problem occurs when many rules cover an example and are labelled with different classes and the non-covering problem when no rule covers the current example and it concerns the choice of the default rule or class.

For the first problem, even if there exist one rule that correctly classifies the example, we are in a misclassification case because, rules covering the example are concurrent and there is no reasonable way to choose (or favour) the rule that correctly classifies the example at the classification step. Using decision lists to deal with such problem could be faithful for some case but it contributes to consider rules in non symmetric way which favours some rules over others.

For the non covering case, most of existing rule based classifiers assigns an example not covered by any rule to the largest class. The choice of the default class is done at the induction time and is not dynamically updated thereafter at the deduction time.

In this work we have proposed a family of Possibilistic Rule-based Classifiers (PRCs) which is based on a Possibilistic Rule-based Reasoning to deal with the two previously presented problems. The PRC is an extension and a modification of the PART classifier basically founded on the sequential covering algorithm and the decision tree principal. The PRC keeps the same rule learning step as the PART and differs mainly in more than one sight. In particular, the PRC learns fuzzy rules instead of crisp rules. Our intuition behind rule fuzzification is that it will help the classifier to more distinguish between rule decision boundaries. To deal with the multiple classification problem, we have proposed to consider weighted rules at classification step in an *unordered* manner instead of rule lists which gives symmetric chance to each rule to be used for classification. Finally for the non covering problem, we intended to *dynamically* assign a default class for examples not covered by any rule using fuzzy rules with large supports. Experiments of the PRCs on a variety of

data sets show their ability to improve the accuracy of the classical PART algorithm.

In the second part of this thesis, we are interested to cope with *uncertainty* in the classification of data with *numerical* attributes. Two common issues are usually countered in this field: i) estimating the distribution associated to numerical attributes which could be the closest representation of the real data and ii) modelling and also handling uncertainty related to attributes and classes when estimating these distributions.

To deal with numerical data, most of classification techniques use a discretization process for continuous attributes and then apply a multinomial probability distribution which leads generally to a loss of information. Other approaches estimate densities in parametric way using Gaussian densities or non parametric way using kernel density functions.

For a long time, Naive Bayesian Classifiers (NBC) has been largely used in a variety of contexts to deal with numerical data. These classifiers, which rely on independence hypotheses, together with a normality assumption to estimate densities for numerical data, are known for their simplicity and their effectiveness.

However estimating densities, even under the normality assumption, may be problematic in case of poor data. Even if a normal distribution is appropriate, identifying it exactly from a sample of data is especially questionable when data are poor. When normality assumption is violated, Gaussian mixtures can be used for approximating any type of distributions. Then, it is required to assess many parameters, a task that may be not compatible with poor data. The problem of the precise estimation of probability distributions for NBCs is important for the exact computation of the probability distribution over the classes. Indeed, a possibility distribution may be viewed as representing a family of probability distributions corresponding to imprecise probabilities, which sound more reasonable in case of poor data. Moreover, we no longer need to assume a particular shape of probability distribution in this possibilistic approximation process.

Thus possibility distributions may provide a more faithful representation of these data. Naive Possibilistic Classifiers (NPC), based on possibility theory, have been recently proposed as a counterpart of Bayesian classifiers to deal with classification task.

The study of possibilistic classifiers is motivated by the good performance of NBCs and by the ability of possibility theory to handle poor data. In spite of the fact that possibility distributions are useful for representing imperfect knowledge, there have been only few works that use Naive Possibilistic Classifiers and most of existing NPC deal only with categorical attributes. This thesis focuses on the estimation of possibility distributions for continuous data. For this reason, we introduce Naive

Possibilistic Classifiers (NPC) that are based on the possibilistic counterpart of the Bayesian formula and the estimation of the possibility distributions. In this work we have developed two kinds of possibilistic classifiers:

The first family called, Gaussian-based Possibilistic Classifiers, assumes normality assumption when estimating possibilistic distributions. For this family of classifiers, we have used two probability-possibility transformation methods enabling us to derive a possibilistic distribution from a probabilistic one. The first method is a direct transformation enabling to move from a classical NBC to a NPC, which introduces some further tolerance in the description of classes. Then, we have tested the feasibility of a Flexible Naive Possibilistic Classifier, which is the possibilistic counterpart of the Flexible Naive Bayesian Classifier. The FNPC estimates possibilistic distributions in a non-parametric way by applying the transformation method to kernel densities instead of Gaussian ones.

Even if we assume that the data follows a Gaussian probabilistic distribution, its parameters are estimated from a limited sample set, and are then necessarily pervaded with imprecision. In the second transformation method, we consider the possibility distribution that encodes all the Gaussian distributions for which the parameters are in a chosen confidence interval. This allows us to generate two new possibilistic classifiers in this family, namely the NPC-2 and the FNPC-2.

The second family of possibilistic classifiers abandons the normality assumption and takes use of a proximity idea between data values in different ways, which provides a less constrained representation of them. We have proposed two other classifiers named Fuzzy Histogram Classifier and Nearest Neighbor-based Possibilistic Classifier in this context. The two proposed classifiers exploit proximity between attribute values in order to estimate possibility distributions. In the first classifier, we compute an average proximity, whereas for the second one we analyze proximities between attributes without counting them. The main advantage of this family of classifiers, when compared to the first one, is their ability to derive possibilistic distributions without the need of a normality assumption, which may lead to a more realistic representation of data.

We have showed that Possibilistic Classifiers have a better capability to detect new instances for which the classification is ambiguous than Bayesian classifiers, where probabilities may be poorly estimated and illusorily precise. In fact, due to the use of the product for combining probability values (which are often small), the errors on probability estimations may have a significant effect on the final estimation. This contrasts with possibility distributions which are less sensitive to imprecise estimation.

Moreover, we have shown that possibilistic classifiers have a higher ability to

detect ambiguity between classes than Bayesian classifiers. Namely the former acknowledge the fact that it is difficult to classify some examples by assessing close possibility degrees to competing classes, whereas the latter in the same situation may give the illusion of discriminating between classes by assessing very different probability degrees to them.

In order to improve the performance of possibilistic classifiers, we have proposed a hybrid Possibilistic Classification approach based on a Nearest Neighbour Heuristics to improve the accuracy of the proposed possibilistic classifiers when the available information is insufficient to choose between classes. The Nearest Neighbor Heuristic contributes to help the main classifier to converge to the correct class label in case data information is insufficient for a more precise classification, rather than choosing between classes having very close plausibility estimates in a rather arbitrary way.

Another significant improvement in accuracy of Gaussian-based possibilistic classifiers is also noted when these latter takes into account priori distribution over classes. The extended version using this modification contributes to significantly improve the accuracy of NPC, FNPC, NPC-2, and FNPC-2 especially for data sets with classes not equivalently covered.

The last main contribution in this thesis is the treatment of uncertainty when classifying numerical data using possibility theory. It should be noted that most of existing works on Naive Possibilistic Classifiers handle uncertainty only for the discrete attributes and are not suitable for numerical attributes. There are some approaches dealing with imperfection in numerical data which exploits probability theory as a tool for this purpose and they cope only with uncertainty in the attribute values without considering uncertainty on class labels. At the best of our knowledge there is no complete work that takes use of possibility theory to cope with imperfection on both attributes and classes in the case of numerical data.

Following this line here, we have intended to exploit Naive Possibilistic classifiers already proposed as a counterpart to Bayesian classifiers to deal with classification task in presence of uncertainty. For this reason, as a second contribution in this part, we extend these classifiers in order to deal with uncertainty in data representation. We consider two types of uncertainty: i) the uncertainty associated to the class in the training set, which is modeled by a possibility distribution over class labels, and ii) the imprecision pervading attribute values in the testing set represented under the form of intervals for continuous data.

First, we have adapted the possibilistic classification model, previously proposed for the certain case, in order to accommodate the uncertainty about class labels. For this case, we have proposed to integrate individual possibility distributions on classes in the calculus of conditional distributions of attributes.

We have also proposed an algorithm based on the extension principle to deal with uncertainty in attribute values. This algorithm seeks to estimate possibility distributions for an uncertain attribute (interval) by looking for possibility distributions of each attribute in the training set belonging to this interval.

In order to test and compare the efficiency of both Possibilistic Classifiers and Possibilistic Rule-based Classifiers, we have implemented all the proposed approaches within the JAVA language using Borland JBuilder environment (10.0.176.120). This choice helps us to integrate available implementation for the PART algorithm from WEKA toolbox. We have conducted several experimentations on a variety of data sets in the perfect or imperfect case. Because of the lack of real imperfect data sets, and in order to test the efficiency of possibilistic classifiers in the imperfect case, we have artificially created imperfect version of each data set with uncertain classes and attributes.

Possibilistic classifiers are compared to classical or flexible Bayesian classifiers on a collection of benchmarks databases. The experiments reported show the interest of possibilistic classifiers. In particular, flexible possibilistic classifiers perform well for data agreeing with the normality assumption, while proximity-based possibilistic classifiers outperform others in the other cases. The hybrid possibilistic classification exhibits a good ability for improving accuracy.

On the other hand, results in the uncertain case show the interest of possibilistic classifiers for handling uncertainty in data. In particular, the Gaussian-based possibilistic classifiers show a high efficiency for dealing with imperfect data and turns to be almost as good as when data are not pervaded with uncertainty.

There are still a few open problems and future lines of research steaming from this study; regarding our proposed rule based classifier, we are planning to orient our research to reinforce optimization of the rule learning step (in the PART algorithm) before starting fuzzification in order to have a more optimized rule set. Second, we are aiming to extend the proposed PRC to support uncertainty in data representation when dealing with numerical data. In this context, the PRR is well suitable to handle such type of data.

With regard to our possibilistic classifiers proposed for numerical data, it would be faithful to seek for other possible representations of possibilistic distributions for numerical data either by applying the probability to possibility transformation to other non parametric densities or by looking for other kinds of proximity measures between numerical data.

For the uncertainty management, it would be intersecting to cope with uncertainty in attribute values both in training and testing set. Estimating conditional distributions from training set including uncertain attributes and classes at the same

time seems to be a more complicated issue to be considered. In addition, in order to really exploit the proposed possibilistic classifiers for uncertain data, it is important to test possibilistic approaches on real uncertain data.

Since real data sets could contain both categorical and numerical attributes, it will be faithful to help possibilistic classifiers to adapt them selves in presence of discrete attributes in training or testing sets. We are thinking to generalize these classifiers to consider both types of data (discrete and continuous).

Appendix A: The PC/PRC Toolbox

A.1 Introduction

In this appendix, we present the software denoted **PC/PRC Toolbox** that we have developed under the JBuilder environment. In this toolbox, we have implemented the four fuzzification approaches of the PART algorithm given in Chapter 4 as well as the different possibilistic classifiers presented in Chapter 5. Principal interfaces will be shown to illustrate how to use our toolbox.

A.2 What the PC/PRC Toolbox do?

The PC/PRC Toolbox enables the user either to use possibilistic rule based classifiers or possibilistic classifiers for certain or uncertain numerical data. In particular, it allows to:

- Train, test and use the standard **PART** algorithm [85].
- Train, test and use the three versions of **Possibilistic Rule based Classifers (PRCs**) as an extension of the PART algorithm as well as its improved version [26].
- Train, test and use Bayesian classifiers: the **NBC** and **FNBC** in the perfect case [107].
- Train, test and use the six proposed possibilistic classifiers: the **NPC, NPC-2, FNPC, FNPC-2, FuHC** and **NNPC** in the certain case [23] [24] where all attributes and classes are certain or in the uncertain case [25] [20].
- Train, test and use the **Nearest Neighbor Heuristic (NNH)** alone or by hybridizing with other possibilistic classifiers [24] .

To test different possibilistic classifiers and compare their efficiency to other well-known classifiers on standard data sets, we have integrated the source code of the pre-processing class from the Weka software [168]. This class allows managing data sets before starting classification. Some statistics on attributes are also given.

We have also integrated and adjusted the source code of the class PART from the Weka software in order to test the PART algorithm. The pre-processing and the PART classes are structured in order to response to the needs of our software.

A.3 Main Menu

In this menu (Figure A.7), the user should start by choosing the data set to use for training and testing possibilistic classifiers by using the **Data** menu. After validating this step, other menus are available:

- The **Possibilistic Rule based Classifier** menu, which allows choosing the possibilistic version of the PART algorithm.
- The **Bayesian Vs Possibilistic Classifier** menu, which allows choosing a possibilistic classifier.

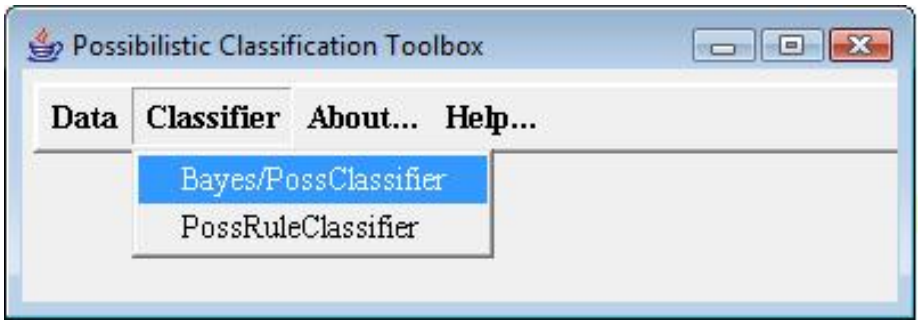

Figure A.7: Possibilistic classification toolbox

A.4 Data Menu

The **Data** menu allows the user to access to the data management dialog box (see Figure A.8), where he/she can open, edit, filter or save a data set file. Through this interface the user can also modify attributes and classes in this data set.

In this dialog box the **Open** button allows to access to other data sets used for experimenting the proposed classifiers as it can be shown in Figure A.9.

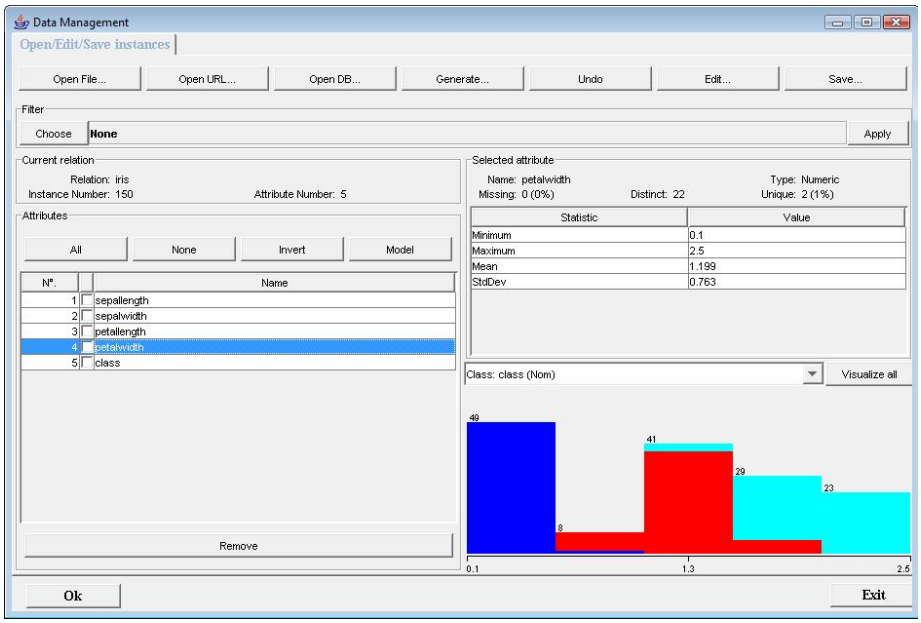

Figure A.8: Data management Dialog

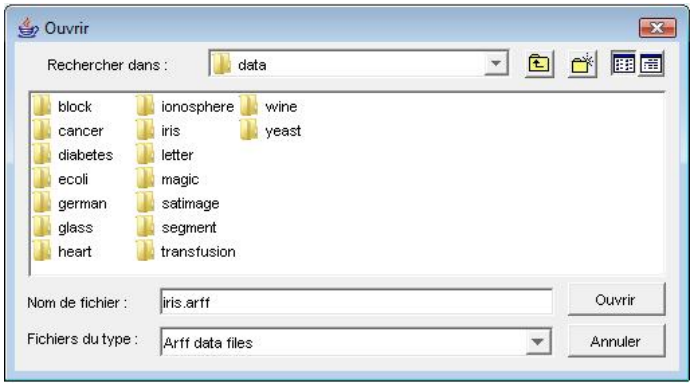

Figure A.9: Datasets used for classification

In our toolbox, we have maintained the same data format, ARFF used in Weka software for coherence. This data format includes two independent parts:

- The *head* part includes the description of attributes (the name and the type) and classes (the distinct values).
- The *data* part includes instances with their respective class values given after "@Data".

An example of an ARFF file is given in Figure A.10 and corresponds to the original iris dataset before normalizing attributes.

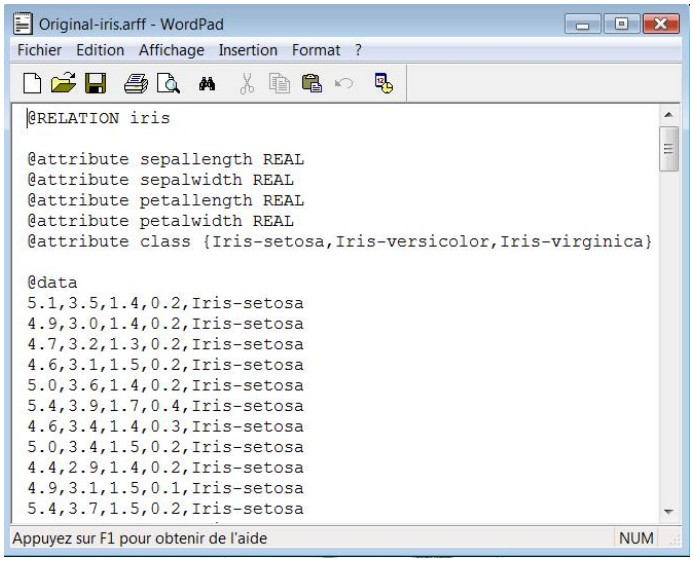

Figure A.10: The content of the original iris file

A.5 Possibilistic Rule based Classifier menu

The menu in Figure A.11 allows to the user training and testing the classical **PART** algorithm as well as the four versions of **Possibilistic Rule based Classifiers** applied to the data set previously chosen in the **Data** menu.

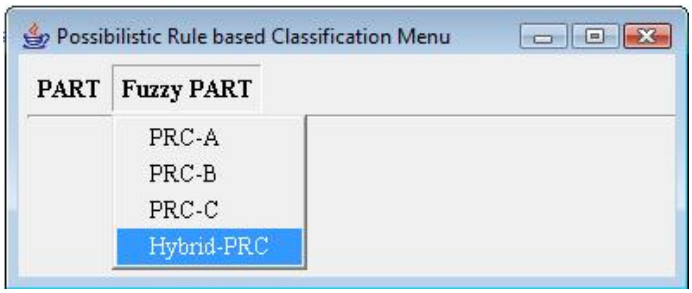

Figure A.11: Possibilistic rule based Classification Menu

A.5.1 Learning and evaluating the PART algorithm

When the user chooses the PART sub-menu, the program will train the PART algorithm on the training set constructed from the original data set and then display the generated rule set. The programm will then proceed to test this rule set on a testing dataset using the 10-cross validation process and display the classification performance of the classifier. Figure A.12 shows an example of a crisp rule set generated by the PART algorithm for the iris data set.

```
Current dataset:     iris
Instances:    150
Attributes:   5
Tested Classifier : PART

---------------------- Classifier Model --------------------

R1:IF(petalwidth<=0.208)===>Class='Iris-setosa'
R2:IF(petalwidth>0.667)===>Class='Iris-virginica'
R3:IF(petallength<=0.678)===>Class='Iris-versicolor'

Number of rules :   3

------------------10-Cross Validation results ----------------

Correctly Classified Instances  141   ===> 94.0±5.54
Incorrectly Classified Instances 9
```

Figure A.12: The PART rule set and the ten-cross validation result for the iris data set

A.5.2 Learning and evaluating the PRCs

1. After selecting the desired Possibilistic Rule based Classifier, the program will first use the same training module of the PART algorithm to generate the crisp rule set.

2. In the second step, the program will execute the **Fuzzification Module** which enables to create a fuzzy version of the crisp rule set using Algorithm 6 of Chapter 4. For this purpose the program will compute and display the lower or upper **Core** and **Support** for each selector in each rule.

3. The last step consists to evaluate the chosen classifier on the testing dataset using the ten-cross validation and finally display the classification performance. Figure A.13 shows the fuzzy version of the crisp rule set generated using the PRC-B and its classification performance.

```
Current dataset:     iris
Instances:    150
Attributes:   5
Tested Classifier : PRC-B

---------------------- Classifier Model --------------------

R1:IF(petalwidth<=0.208)===>Class='Iris-setosa'
R2:IF(petalwidth>0.667)===>Class='Iris-virginica'
R3:IF(petallength<=0.678)===>Class='Iris-versicolor'

Number of rules :   3

--------------------- Fuzzy Rule Set -------------------------

      Selector         Core-U    Core-L      Supp-U       Supp-L
(petalwidth<=0.208)    0.167       -          0.249         -
(petalwidth>0.667)       -       0.708         -          0.626
(petallength<=0.678)   0.576       -          0.78          -

------------------- 10-Cross Validation results --------------

Correctly Classified Instances  145   ===> 96.67±4.47
Incorrectly Classified Instances 5
```

Figure A.13: The Fuzzy rule set generated by the PRC-B and its ten-cross validation result for the iris data set

A.6 Bayesian Vs Possibilistic Classifier menu

This menu allows user to experiment Bayesian or Possibilistic classifiers described in this thesis. Figure A.14 shows the Bayesian panel which includes the **Naive Bayesian classifier (NBC)** and the **Flexible Naive Bayesian Classifier (FNBC)**. The Possibilistic Panel is illustrated through Figure A.15.

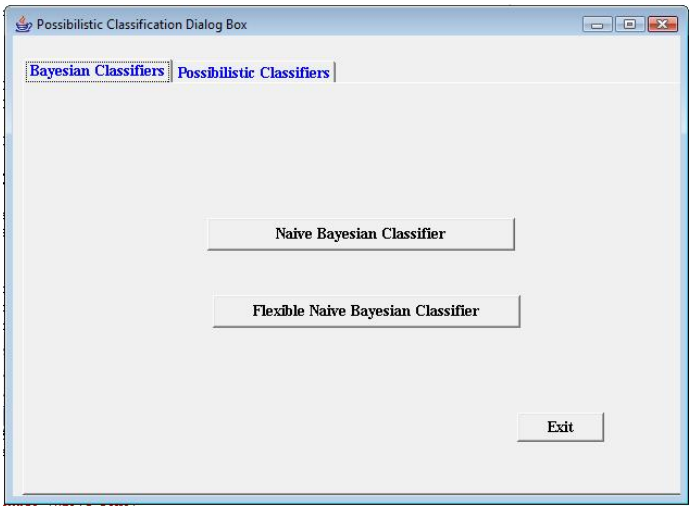

Figure A.14: Bayesian Classification Panel

A.6.1 Possibilistic Classification Panel

This dialog box allows experimenting Possibilistic Classifiers as well as the Nearest Neighbor Heuristic. Before training and testing a classifier, the user is asked to specify two principal classification parameters:

1. In the *Hybridizing* panel, one can choose to hybridize or not the current classifier with the Nearest Neighbor Heuristic to improve its efficiency as introduced in Section 5.4.3. Moreover this panel allows testing the NNH considered here as an independent classifier by using the **Nearest Neighbor Heuristic** button.

2. In the *Uncertainty* panel the user is asked to choose the type of uncertainty to deal with for the current classifier, especially uncertainty about classes or attributes. When the user chooses the uncertain case (for classes or attributes), he/she should generate the corresponding data using the **Generate Uncertain Data** button before doing classification.

After fixing classification parameters, the user can test and use one of the six proposed possibilistic classifiers, namely, the **NPC**, **NPC-2**, the **FNPC**, the **FNPC-2**,

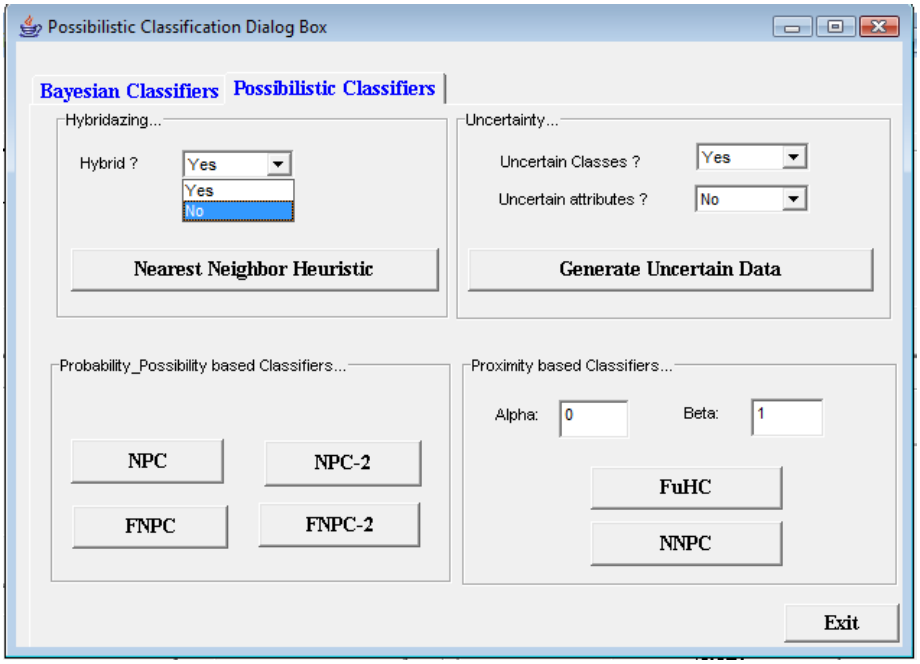

Figure A.15: Possibilistic Classification Panel

the **FuHC** or the **NNPC**. For the two proximity based possibilistic classifiers, two supplementary parameters are asked to be fixed, *alpha* and *beta* which are necessary to compute the proximity relation given in Equation 5.10.

The program will then start training and testing the selected classifier on the current data set and then display results of the two evaluation criteria, *MPcc* and *AffC* over the ten-cross validation as shown in Figure A.16.

```
--------------- Possibilistic Classifiers --------------

Current Data set:   Certain-iris
Tested classifier:   FNPC

------------- 10-Cross Validation results --------------

 Mean MPcc (%):    96.0±4.42
 Mean AffC :       0.97±0.03
```

Figure A.16: Results of the ten-cross validation for the FNPC applied to the iris dataset

A.6.2 Uncertainty dialog box

When we choose to deal with uncertainty (using The Generate Uncertain Data Button), the dialog box in Figure A.17 appears where the user is asked to choose the uncertainty rate on attributes and/or classes before generating the uncertain data

corresponding to that chosen in the **Data Management dialog**. Different uncertainty rates are proposed (0, 0.25, 0.5, 0.75, 1) for each uncertainty type. In particular, the 0-rate corresponds the total certain version whereas 1-rate corresponds to the case where all instances or attributes are assumed to be uncertain.

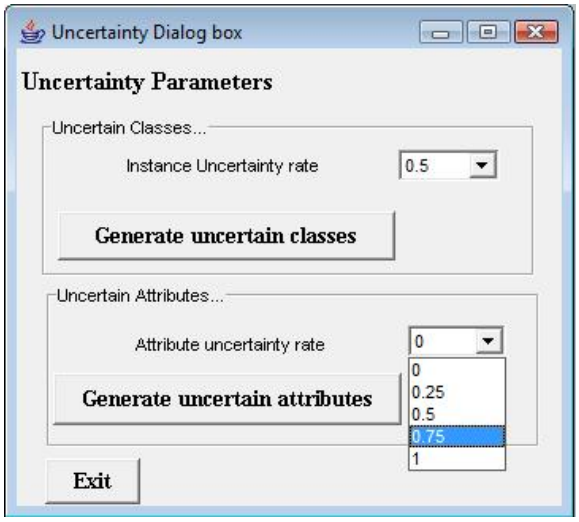

Figure A.17: Data management

As a result, an uncertain file version of the original data set is created. The Figure A.18 and A.19 show respectively the content of uncertain iris training and testing files. In the uncertain training file, uncertainty concerns the class where each instance includes a possibility distribution on all possible class labels. In the iris testing set, attributes are replaced by an interval to illustrate imprecision on numerical values.

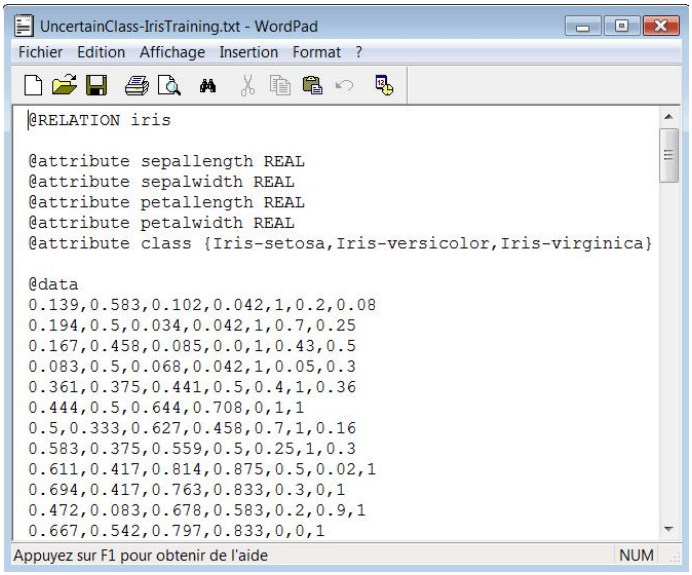

Figure A.18: The content of uncertain iris training file

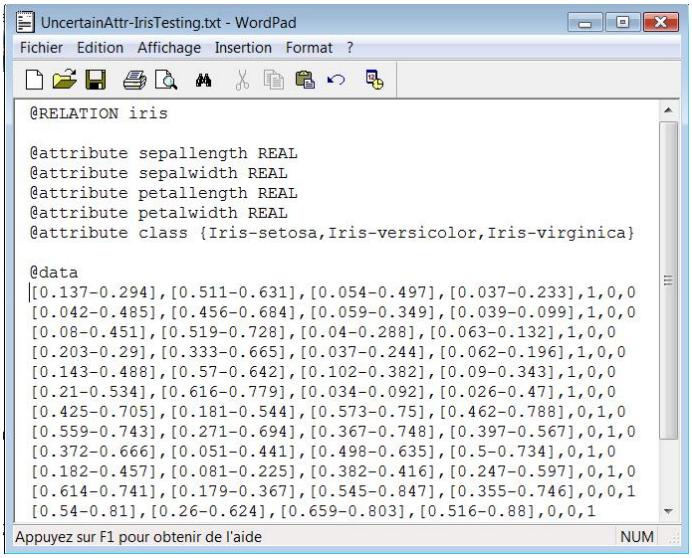

Figure A.19: The content of uncertain iris testing file

A.7 Conclusion

In this appendix we have presented our toolbox denoted **PC/PRC Toolbox** which enables to train and test all possibilistic classification approaches proposed in this thesis. We have illustrated some dialog boxes and menus in this user guide that explain the principal roles of this toolbox. In particular, we have shown the format of the input data as well as the format of the result displayed to the user for each classifier.

The implemented toolbox, by its generic nature, can be tested on any other numerical data set that conserves the "ARFF" format and can be easily integrated in any other application developed in this environment.

Bibliography

[1] D.W. Aha. *Lazy Learning*. Kluwer Academic Publishers, Dordrecht, 1997.

[2] A. Aijun and C. Nick. Rule quality measures improve the accuracy of rule induction: An experimental approach. *In Proceding of the International Syposium on Methodologies for Intelligent Systems*, pages 119–129, 2000.

[3] N. Ben Amor, S. Benferhat, and Z. Elouedi. Qualitative classification and evaluation in possibilistic decision trees. *In Proceding of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2004)*, 1:653–657, 2004.

[4] N. Ben Amor, S. Benferhat, K. Mellouli, and S. Smaoui. Inférence dans les réseaux possibilistes basés sur le conditionnement ordinal. *Revue d'Intelligence Artificielle*, pages 489–519, 2007.

[5] N. Ben Amor, K. Mellouli, S. Benferhat, D. Dubois, and H. Prade. A theoretical framework for possibilistic independence in a weakly ordered setting. *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, 10:117–155, 2002.

[6] A. Aregui and T. Denœux. Consonant belief function induced by a confidence set of pignistic probabilities. *In Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2007)*, 4724:344–355, 2007.

[7] B.C. Arnold and R.M. Shavelle. Joint confidence sets for the mean and variance of a normal distribution. *The American Statistician*, 52(2):133–140.

[8] S. Benferhat, D. Dubois, L. Garcia, and H. Prade. On the transformation between possibilistic logic bases and possibilistic causal networks. *International Journal of Approximate Reasoning*, 29(2):135–173, 2002.

[9] S. Benferhat, D. Dubois, and H. Prade. Representing default rules in possibilistic logic. *In Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KRR 1992)*, pages 673–684, 1992.

[10] S. Benferhat and K. Tabia. An efficient algorithm for naive possibilistic classifiers with uncertain inputs. *In Proceding of the 2nd International Conference on Scalable Uncertainty Management (SUM 2008)*, 5291:63–77, 2008.

[11] H. R. Berenji. The treatment of uncertainty in artificial intelligence. *Machine Intelligence for aerospace robotics*, 1988.

[12] J. Beringer and E. Hüllermeier. Case-based learning in a bipolar possibilistic framework. *International journal of intelligent systems*, 23:1119–1134, 2008.

[13] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York USA, 1996.

[14] C.M. Bishop. Latent variable models. *Learning in Graphical Models*, pages 371–403, 1999.

[15] P. Bonissone and R. M. Tong. Editorial reasoning with uncertainty in expert systems. *International journal of Man Machine Studies*, 22:241–250, 1985.

[16] C. Borgelt and J. Gebhardt. A naïve bayes style possibilistic classifier. *In Proceedings of the 7th European Congress on Intelligent Techniques and Soft Computing*, pages 556–565, 1999.

[17] C. Borgelt, J. Gebhardt, and R. Kruse. Possibilistic graphical models. *In Proceedings of the International School for the Synthesis of Expert Knowledge*, pages 51–68, 1998.

[18] C. Borgelt and R. Kruse. Efficient maximum projection of database-induced multivariate possibility distributions. *In Proceding of the 7th IEEE International Conference on Fuzzy Systems*, pages 663–668, 1998.

[19] P. Bosc, D. Dubois, and H. Prade. An introduction to fuzzy sets and possibility theory based approaches to the treatment of uncertainty and imprecision in database management systems. *In Proceedings of the second Workshop on Uncertainty Management in Information Systems : From needs to solutions*, pages 44–70, 1993.

[20] M. Bounhas, M.H. Ghasemi, H. Prade, M. Serrurier, and K. Mellouli. Naive possibilistic classifiers for imprecise or uncertain numerical data. *Fuzzy Sets and Systems Journal*, Special Issue of ECSQARU 2011, 2012 (In first revision).

[21] M. Bounhas and K. Mellouli. A possibilistic classification approach to handle continuous data. *In Proceding of the 8th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2010)*, pages 1–8, 2010a.

[22] M. Bounhas and K. Mellouli. Using class-based reasoning to improve the accuracy of symbolic rules in a hybrid possibilistic approach. *In Proceding of*

The 2th International Conference on Advances in Databases Knowledge and Data Applications, pages 222–228, 2010b.

[23] M. Bounhas, K. Mellouli, H. Prade, and M. Serrurier. From bayesian classifiers to possibilistic classifiers for numerical data. *In Proceding of The 4th International Conference on Scalable Uncertainty Management (SUM 2010)*, LNAI 6379:112–125, 2010.

[24] M. Bounhas, K. Mellouli, H. Prade, and M. Serrurier. Possibilistic classifiers for numerical data. *In Soft Computing Journal*, 2012 (to appear).

[25] M. Bounhas, H. Prade, M. Serrurier, and K. Mellouli. Possibilistic classifiers for uncertain numerical data. *In Proceding of the 11th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (EC-SQARU 2011)*, LNAI 6717:434–446, 2011.

[26] M. Bounhas, H. Prade, M. Serrurier, and K. Mellouli. A possibilistic rule-based classifier. *In Proceding of the 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2012), S. Greco et al. (Eds.), Part I, CCIS 297*, pages 21–31, 2012.

[27] L. Breiman, J. Friedman, R. Olsen, and C. Stone. *Classification and Regression Trees*. Chapman and Hall, London UK, 1984.

[28] H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, 6:153–172, 2002.

[29] C.A. Brink and M.J. Pazzani. An investigation of noise-tolerant relational concept learning algorithms. *In Proceding of the 8th International Workshop on Machine Learning*, pages 389–393, 1991.

[30] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:1–47, 1998.

[31] L.S. Camargo and T. Yoneyama. Specification of training sets and the number of hidden neurons for multilayer perceptrons. *Neural Computation*, 13:2673–2680, 2001.

[32] E. Castineira, S. Cubillo, and E. Trillas. On the coherence between probability and possibility measures. *International Journal of Information Theories and Applications*, 14, 2007.

[33] J. Cheng and R. Greiner. Comparing bayesian network classifiers. *In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 101–107, 1999.

[34] M.R. Civanlar and H.J. Trussel. Constructing membership functions using statistical data. *Fuzzy Sets and Systems*, 18(1):1–13, 1986.

[35] M.R. Civanlar and H.J. Trussell. Constructing membership functions using statistical data. *Fuzzy Sets and Systems*, 18:1–13, January 1986.

[36] P. Clark and R. Boswell. Rule induction with cn2: Some recent improvements. *Machine Learning*, pages 151–163, 1991.

[37] P. Clark and T. Niblett. The cn2 induction algorithm. *Machine Learning Journal*, 3(4):261–283, 1989.

[38] W.W. Cohen. Efficient pruning methods for separate and conquer rule learning systems. *In Proceding of the 13th International Joint Conference on AI*, pages 988–994, 1993.

[39] W.W. Cohen. Fast effective rule induction. *In Proceding of the 12th International Conference on Machine Learning*, pages 115–123, 1995.

[40] G. De Cooman. Possibility theory. part i: Measure- and integral-theoretic ground-work; part ii: Conditional possibility; part iii: Possibilistic independence. *International Journal of General System*, 25:291–371, 1997.

[41] G. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

[42] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):7–21, 1967.

[43] T.M. Cover and P.E. Hart. Nearest neighbour pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[44] B.V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Comp. Soc. Press, 1991.

[45] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. *In Proceding of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 211–219, 1996.

[46] K.A. DeJong, W.M. Spears, and D.F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 13:161–188, 1993.

[47] M. Delgado. On the concept of possibility-probability consistency. *Fuzzy Sets and Systems*, 21(3):311–318, 1987.

[48] M. Delgado and S. Moral. On the concept of possibility-probability consistency. *Fuzzy Sets and Systems*, 21(3):311–318, 1987.

[49] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, 38:325–339, 1967.

[50] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[51] T. Denœux and L.M. Zouhal. Handling possibilistic labels in pattern classification using evidential reasoning. *Fuzzy set and systems*, 122(3):47–62, 2001.

[52] A. Denton and W. Perrizo. A kernel-based semi-naive bayesian classifier using p-trees. *In Proceding of the 4th SIAM International Conference on Data Mining*, pages 427–431, 2004.

[53] L. Devroye. The equivalence of weak strong and complete convergence in l1 for kernel density estimates. *The Annals of Statistics*, 11:896–904, 1983.

[54] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. *Machine Learning*, 29:102–130, 2002.

[55] D. Dubois. Possibility theory and statistical reasoning. *Computational Statistics and Data Analysis*, 51:47–69, 2006.

[56] D. Dubois, D. Cayrac, and H. Prade. Handling uncertainty with possibility theory and fuzzy sets in a satellite fault diagnosis application. *In Proceedings of the 6th International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems*, pages 251–269, 1996.

[57] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. *In Handbook of Logic in Artificial Intelligence and Logic Programming*, 3:439–513, 1994.

[58] D. Dubois, F. Laurent, M. Gilles, and H. Prade. Probability-possibility transformations triangular fuzzy sets and probabilistic inequalities. *Reliable Computing*, 10:273–297, 2004.

[59] D. Dubois, H.T. Nguyen, and H. Prade. Possibility theory probability and fuzzy sets : Misunderstandings bridges and gaps. *Fundamentals of Fuzzy sets*, pages 343–438, 2000.

[60] D. Dubois and H. Prade. On several representations of an uncertain body of evidence. *Fuzzy Information and Decision Process*, pages 167–181, 1982.

[61] D. Dubois and H. Prade. A set-theoretic view of belief functions : logical operations and approximations by fuzzy sets. *International Journal of General Systems*, 12(3):193–226, 1986.

[62] D. Dubois and H. Prade. *Théorie des Possibilités: Application a la Représentation des Connaissances en Informatique*. MASSON, Paris France, 1987.

[63] D. Dubois and H. Prade. *Possibility Theory: An Approach to computerized Processing of Uncertainty*. Plenum Press, 1988.

[64] D. Dubois and H. Prade. Aggregation of possibility measures. *Multiperson Decision Making using Fuzzy Sets and Possibility Theory*, pages 55–63, 1990.

[65] D. Dubois and H. Prade. The logical view of conditioning and its application to possibility and evidence theories. *International Journal of Approximate Reasoning*, 4(1):23–46, 1990.

[66] D. Dubois and H. Prade. When upper probabilities are possibility measures. *Fuzzy sets and systems*, 49(1):65–74, 1992.

[67] D. Dubois and H. Prade. Fuzzy sets and probability : Misunderstandings bridges and gaps. *In Proceedings of the 2d IEEE International Conference of Fuzzy Systems (FUZZ-IEEE)*, pages 1059–1068, 1993.

[68] D. Dubois and H. Prade. On data summarization with fuzzy sets. *In Proceding of the 5th International Fuzzy Systems Assoc. World Congress (IFSA93)*, 1993.

[69] D. Dubois and H. Prade. An introductory survey of possibility theory and its recent developments. *Journal of Japan Society for Fuzzy Theory and Systems*, 10:21–42, 1998.

[70] D. Dubois and H. Prade. Possibility theory: Qualitative and quantitative aspects. *Handbook on Defeasible Reasoning and Uncertainty Management Systems*, 1:169–226, 1998a.

[71] D. Dubois and H. Prade. An overview of ordinal and numerical approaches to causal diagnostic problem solving. *Abductive Reasoning and Learning Handbooks of Defeasible Reasoning and Uncertainty Management Systems Drums Handbooks*, 4:231–280, 2000.

[72] D. Dubois and H. Prade. Possibilistic logic: a retrospective and prospective view. *Fuzzy Sets and Systems*, 144:3–23, 2004.

[73] D. Dubois and H. Prade. Formal representations of uncertainty. *Decision-making - Concepts and Methods*, pages 85–156, 2009.

[74] D. Dubois, H. Prade, and S. Sandri. On possibility/probability transformations. *Fuzzy Logic - State of the Art*, pages 103–112, 1993.

[75] Didier Dubois. The role of epistemic uncertainty in risk analysis. *In Proceedings of the 4th international conference on Scalable uncertainty management (SUM 2010)*, pages 11–15, 2010.

[76] W. Duch, R. Setiono, and J. Zurada. Computational intelligence methods for rule-based data understanding. *In Proceeding of the IEEE*, 92(5):771–805, 2004.

[77] M. Eineborg and H. Boström. Classifying uncovered examples by rule stretching. *In Proceedings of the 11th International Conference of Inductive Logic Programming (ILP 2001)*, pages 41–50, 2001.

[78] P. Fabiani. *Représentation Dynamique de l'Incertain et Stratégie de Prise d'Information Pour un Système Autonome en Environnement Evolutif*. PhD thesis in Automatic and Industriel Informatics Ecole Nationale Supérieure de l'Aéronautique et de l'Espace. Toulouse, 1996.

[79] T. Fawcett. Prie: a system for generating rulelists to maximize roc performance. *Data Mining and Knowledge Discovery*, 17:207–224, 2008.

[80] M. Figueiredo and J.M.N. Leit ao. On fitting mixture models. *In Energy Minimization Methods in Computer Vision and Pattern Recognition*, 1654:732–749, 1999.

[81] D. B. Finetti. La prévision: Ses lois logiques ses sources subjectives. *Annales de l'Institut Henri Poincaré*, 7, 1937.

[82] J. Fodor and R. Yager. Fuzzy set-theoretic operators and quantifiers. *Fundamentals of Fuzzy Sets The Handbook of Fuzzy Sets Series*, pages 125–193, 2000.

[83] P. Fonck. Réseaux d'inférence pour le raisonnement possibiliste, 1994.

[84] O. François and P. Leray. Etude comparative d'Algorithmes d'Apprentissage de Structure dans les Réseaux Bayésiens. *Journal électronique d'intelligence artificielle*, 5(39):1–19, 2004.

[85] E. Frank and I.H. Witten. Generating accurate rule sets without global optimization. *In Proceding of the 15th International Conference in Machine Learning*, pages 144–151, 1998.

[86] A.A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag, 2002.

[87] J.H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 1989.

[88] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–161, 1997.

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

[90] J. Fürnkranz and G. Widmer. Incremental reduced error pruning. *In Proceedings of the 11th International Conference on Machine Learning (ML 1994)*, pages 70–77, 1994.

[91] D. Geiger and D. Heckerman. Learning gaussian networks. 1994. Available as Technical Report MSR-TR-94-10.

[92] D. Grossman and P. Dominigos. Learning bayesian maximizing conditional likelihood. *In Proceding on Machine Learning*, pages 46–57, 2004.

[93] B. Haouari, N. Ben Amor, Z. Elouadi, and K. Mellouli. Naïve possibilistic network classifiers. *Fuzzy Set and Systems*, 160(22):3224–3238, 2009.

[94] I. Hatzilygeroudisa and J. Prentzas. Integrating rules neural networks and cases for knowledge representation and reasoning in expert systems. *Expert Systems with Applications*, 27:63–75, 2004.

[95] I. Hatzilygeroudisa and J. Prentzas. Neuro-symbolic approaches for knowledge representation in expert systems. *International Journal of Hybrid Intelligent Systems*, 1(3-4):111–126, 2004.

[96] R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[97] J. Hühn and E. Hüllermeier. Furia: An algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19:293–319, 2009.

[98] E. Hüllermeier. Possibilistic instance-based learning. *Artificial Intelligence*, 148(1-2):335–383, 2003.

[99] E. Hüllermeier. Fuzzy methods in machine learning and data mining : Status and prospects. *Fuzzy Sets and Systems*, 156(3):387–406, 2005.

[100] E. Hunt, J. Martin, and P. Stone. *Experiments in Induction*. Academic Press, New York USA, 1966.

[101] H. Ishibuchi and T. Yamamoto. Rule weight specification in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 13:428–436, 2005.

[102] I. Jenhani. *From Possibilistic Similarity Measures to Possibilistic Decision Trees*. PhD Thesis in The Higher Institute of Management of Tunis (Tunisia) and Artois University (France), 2010.

[103] I. Jenhani, N. Ben Amor, and Z. Elouedi. Decision trees as possibilistic classifiers. *International Journal of Approximate Reasoning*, 48(3):784–807, 2008.

[104] I. Jenhani, S. Benferhat, and Z. Elouedi. Learning and evaluating possibilistic decision trees using information affinity. *International Journal of Computer Systems Science and Engineering*, 4(3):206–212, 2010.

[105] F. Jensen. *An introduction to Bayesian Networks*. UCL Press, 1986.

[106] F.V. Jensen and S.L. Lauritzen. Bayesian updating in recursive graphical models by local computations. *Computational Statistical Quaterly*, 4:269–282, 1990.

[107] G.H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. *In Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 338–345, 1995.

[108] M.I. Jordan and Y. Weiss. *Probabilistic inference in graphical models.* Handbook of neural networks and brain theory Citeseer, 2002.

[109] J.H. Kim and J. Pearl. A computational model for combined causal and diagnostic reasoning in inference systems. *In Proceedings of IJCAI-83*, pages 190–193, 1983.

[110] E.P. Klement, R. Mesiar, and E. Pap. *Triangular norms.* Kluwer Academic Pub., Boston USA, 2000.

[111] G.J. Klir. A principle of uncertainty and information invariance. *International Journal of General Systems*, 17(2-3):249–275, 1990.

[112] I. Kononenko. Semi-naive bayesian classifier. *In Proceding of the European Working Session on Machine Learning*, pages 206–219, 1991.

[113] I. Kononenko and I. Bratko. Information-based evaluation criterion for classifier's performance. *Machine Learning*, 6:67–80, 1991.

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

[115] M. Kubat and M. Cooperson. A reduction technique for nearest-neighbor classification: Small groups of examples. *Intell. Data Anal.*, 5(6):463–476, 2001.

[116] W. Van Laer and L. De Raedt. How to upgrade propositional learners to first order logic : A case study. *Relational Data Mining*, pages 235–261, 2001.

[117] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. *In Proceedings of AAAI-92*, 7:223–228, 1992.

[118] P. Langley and S. Sage. Induction of selective bayesian classifiers. *In Proceding of the 10th Conference on Uncertainty in Artificial Intelligence (UAI 1994)*, pages 399–406, 1994.

[119] V. Lasserre, G. Mauris, and L. Foulloy. A simple possibilistic modelisation of measurement uncertainty. *Uncertainty in Intelligent and Information Systems*, pages 58–69, 2000.

[120] S. Lauritzen. Local computation with probabilities on graphical structures and their application to expert systems. *In Journal of the Royal Statistical Society*, 50:157–224, 1988.

[121] B. Lavoie. Apprentissage automatique de règles. *Programme de Doctorat en Informatique Cognitive Université du Québec à Montréal*, 2006.

[122] R.P. Lipmann. An introduction to computing with neural nets. *IEEE ASSP magazine*, pages 4–21, 1987.

[123] W. Liu. Conflict analysis and merging operators selection in possibility theory. *In Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 816–827, 2007.

[124] D. Mackay. An introduction to monte carlo methods. *In Learning in Graphical Models MIT Press*, pages 175–204, 1999.

[125] K. Kalyani Manchi and X. Wu. Dynamic refinement of classification rules. *In Proceding of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2002)*, page 189, 2002.

[126] R. Lopez De Mantaras and E. Armengol. Machine learning from examples: Inductive and lazy methods. *Data and Knowledge Engineering*, 25:99–123, 1998.

[127] P. Martin-Munoz and F.J. Moreno-Velo. Fuzzycn2: An algorithm for extracting fuzzy classification rule lists. *In Proceding of the IEEE International Conference of Fuzzy Systems (FUZZ-IEEE) Barcelona Spain*, pages 1–7, 2010.

[128] M.H. Masson. Apports de la théorie des possibilités et des fonctions de croyance à lanalyse de données imprécises, 2005.

[129] G.J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics, 2000.

[130] S. Meganck, Ph. Leray, S. Maes, and B. Manderick. Apprentissage des réseaux bayésiens causaux à partir de données d'observation et d'expérimentation. *In 15e congrès francophone AFRIF-AFIA Reconnaissance des Formes et Intelligence Artificielle*, 2006.

[131] J. Mertz and P.M. Murphy. Uci repository of machine learning databases. *Available at: ftp://ftp.ics.uci.edu/pub/machine-learning-databases*, 2000.

[132] R.S. Michalski. On the quasi-minimal solution of the general covering problem. *In Proceding of the 5th International Symposium on Information Processing*, pages 125–128, 1969.

[133] R.S. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning system aq15 and its testing application to three medical domains. *In Proceedings of the AAAI*, pages 1041–1045, 1986.

[134] T.M. Mitchell. *Machine Learning*. McGraw-Hill, New York USA, 1997.

[135] S. Muggleton and C. Feng. Efficient induction of logic programs. *In Proceedings of the 1st Conference on Algorithmic Learning Theory*, pages 368–381, 1990.

[136] S. Okamoto and N. Yugami. Effects of domain characteristics on instance-based learning algorithms. *Theoretical Computer Science*, 298:207–233, 2003.

[137] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning Journal*, 5:71–100, 1990.

[138] J. Pearl. Reverand bayes on inference engines: A distributed hierarchical approach. *In Proceedings of the AAAI National Conference on AI*, pages 133–136, 1982.

[139] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmman, San Francisco CA USA, 1988.

[140] A. Pérez, P. Larra naga, and I. Inza. Supervised classification with conditional gaussian networks: Increasing the structure complexity from naive baye. *International Journal of Approximate Reasoning*, 43:1–25, 2006.

[141] A. Pérez, P. Larra naga, and I. Inza. Bayesian classifiers based on kernel density estimation:flexible classifiers. *International Journal of Approximate Reasoning*, 50:341–362, 2009.

[142] J. Prentzas and I. Hatzilygeroudis. Categorizing approaches combining rule-based and case-based reasoning. *Expert Systems*, 24:97–122, 2007.

[143] B. Qin, Y. Xia, and F. Li. A bayesian classifier for uncertain data. *In Proceding of the 25th ACM Symposium on Applied Computing (SAC)*, pages 1010–1014, 2010.

[144] B. Qin, Y. Xia, S. Prabhakar, and Y. Tu. A rule-based classification algorithm for uncertain data. *In Proceding of the IEEE International Conference on Data Engineering*, 2009.

[145] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Fransisco USA, 1993.

[146] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[147] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

[148] J.R. Quinlan. Mdl and categorical theories. *In Proceedings of International Conference on Machine Learning*, pages 464–470, 1995.

[149] J.R. Quinlan and R.M. Cameron-Jones. Foil : a midterm report. *In Proceedings of the European Conference on Machine Learning*, pages 3–20, 1993.

[150] J.R. Quinlan and R.L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation/information and Control*, 80:227–248, 1989.

[151] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1:318–362, 1986.

[152] M. Sahami. Learning limited dependence bayesian classifiers. *In Proceding of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 335–338, 1996.

[153] S. Sandri. *La Combinaison de l'Information Incertaine et ses Aspects Algorithmiques*. Thèse de Doctorat d'Informatique. Université Paul Sabatier de Toulouse, 1991.

[154] L. J. Savage. *The Foundations of Statistics*. Dover Publications, New York USA, 1972.

[155] M. Serrurier and H. Prade. Coping with exceptions in multiclass ilp problems using possibilistic logic. *In Proceding of the International Joint Conference on Artificial Intelligence IJCAI 2005*, pages 1761–1764, 2005.

[156] G. Shafer. *A mathematical theory of evidence*. Princeton University Press, Princeton New Jersey USA, 1976.

[157] Ph. Smets. Imperfect information : Imprecision - uncertainty. *Uncertainty Management in Information Systems*, pages 225–254, 1997.

[158] Ph. Smets. The application of the transferable belief model to diagnostic problems. *International Journal of Intelligent Systems*, 13:127–157, 1998.

[159] Ph. Smets and R. Kennes. The transferable belief model. *Artificial Intelligence*, 66:191–234, 1994.

[160] R. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7:224–254, 1964.

[161] O. Strauss, F. Comby, and M.J. Aldon. Rough histograms for robust statistics. *In Proceding of the International Conference on Pattern Recognition (ICPR 2000)*, II:2684–2687, 2000.

[162] T. Sudkamp. Similarity as a foundation for possibility. *In Proceding of the 9th IEEE International Conference on Fuzzy Systems*, pages 735–740, 2000.

[163] H. Sun and M. Farooq. Conjunctive and disjunctive combination rules of evidence. *Signal Processing Sensor Fusion and Target Recognition XIII*, 5429:392–401, 2004.

[164] V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 1995.

[165] P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, 1991.

[166] P. Walley and T. Fine. Towards a frequentist theory of upper and lower probability. *The Annals of Statistics*, 10:741–761, 1982.

[167] D. Wettschereck, D.W. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 10:1–37, 1997.

[168] H.I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. 2nd edition Morgan Kaufmann Publishers, 2005.

[169] M.L. Wong and K.S. Leung. *Data Mining using Grammar Based Genetic Programming and Applications*. Kluwer Academic Publishers, 2000.

[170] J. Yam and W. Chow. Feedforward networks training speed enhancement by optimal initialization of the synaptic coefficients. *IEEE Transactions on Neural Networks*, 12:430–434, 2001.

[171] K. Yamada. Probability-possibility transformation based on evidence theory. *IFSA World Congress and 20th NAFIPS International Conference 2001. Joint 9th*, pages 70–75, 2001.

[172] Y. Yang and G.I. Webb. Discretization for naive-bayes learning: Managing discretization bias and variance. *Technical Report 2003-131*, 2003.

[173] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[174] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

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

[176] H. Zhang. The optimality of naive bayes. *In Proceedings of 17th International FLAIRS Conference (FLAIRS2004)*, 2004.

[177] N.L. Zhang and D. Poole. A simple approach to bayesian network computations. *In Proceding of the 10th Canadian Conference on Artificial Intelligence*, 71:171–178, 1994.

[178] J.M. Zurada. *Introduction to Artificial Neural Systems*. PWS Publishing, Boston MA, 1992.