Intelligent Systems Group

informatika fakultatea | facultad de informática

Universidad del País Vasco | Euskal Herriko Unibertsitatea

Konputazio Zientziak eta Adimen Artifizialaren Saila

Departamento de Ciencias de la Computación e Inteligencia Artificial

Advances in Error Estimation and Multi-Dimensional Supervised Classification

by

Juan Diego Rodríguez

Supervised by Jose A. Lozano and Aritz Pérez

Dissertation submitted to the Department of Computer Science and Artificial Intelligence of the University of the Basque Country (UPV/EHU) as partial fulfilment of the requirements for the PhD degree in Computer Science

Donostia - San Sebastián, January 2013

To my parents. Vuestro apoyo es fundamental para mi.

To Aintzane. Itsasertzean elkartuko gara.

Acknowledgments

The present dissertation would never have been possible without the support of so many people that I hope not to forget anybody. First of all, I have to thank my thesis advisors Aritz Pérez and Jose Antonio Lozano. They guided me accross these years, taught me how to research and converted me in the researcher I am. Their patiente, time, work and encouragement have been fundamental to carry out this project. Moreover, I honestly feel very lucky to have been under the supervision of such honest and good people.

I cannot forget my colleagues in the Intelligent Systems Group with whom I have shared the last years: Iñaki Inza, Alex Mendiburu, Jose Miguel Blanco, Pedro Larrañaga, Carlos Echegoyen, Dinora Morales, Ekhiñe Irurozki, Guzmán Santafé, Jonathan Ortigosa, Jose Antonio Fernándes, Jose Antonio Pascual, Jose Luis Flores, Leticia Hernando, Ramón Sagarna, Roberto Santana, Rubén Armañanzas, Josu Ceberio, Unai López, Tania Lorido, Eneko Mateo, Carlos Pérez, Jerónimo Hernandez, Usue Mori and Javier Navaridas (and Joseba Esteban, not in the group but always flying around). The atmosphere of the laboratory was extraordinary, I consider you friends, not only colleagues. Intentionally, I have not mentioned Borja Calvo. The reason is that I would like to specially thank Borja for his help. He has been a kind of advisor for me, he guided me disinterestedly through this journey and taught me all the tools I needed to carry out my work. Thank you Borja, I doubt I could have reached here without your help.

I am also very grateful to Petri Myllymäki and the whole COSCO group at the Helsinki Information Technology Institute (HIIT) of the Helsinki University. They were exceptional hosts during my stay as a visitor researcher in Finland. It was an unforgettable experience.

Acknowledgments to the Department of Computer Science and Artificial Intelligence and the Ministerio de Educación y Ciencia for the financial support.

If there is anybody that I should be thankful to that is my family. Without the education, encouragement and unconditional support of my parents during my life I would not be here. They have always believed in me and I have always tried to make them proud of me. I am very proud of them. I owe everything to you, I love you. *A mis padres: sin la educación que me habéis brindado, vuestros ánimos y vuestro apoyo incondicional no habría llegado hasta aqui. Siempre habéis creido en mi y yo siempre he tratado de que estéis orgullosos de mi. Yo estoy muy orgulloso de vosotros. Os lo debo todo, os quiero.*

Last but not least, the most special thanks are for Aintzane Tolosa (and the two little devils that came from the east bringing us so much happiness: Nora Yuanzhi and Adi Xuan). She encouraged me to begin this exciting voyage, she always had a smile in the bad days and gave me great inspiration when I was ran out of ideas. Your smile is the lighthouse I was looking for. Tzin, you are my friend, my partner, my lover... Thank you very much for all you gave me. Beti zure ondoan, maite zaitut.

Contents

List of Figures

List of Tables

Computer science is not about machines, in the same way that astronomy is not about telescopes.
–***Michael R. Fellows***–

Life is a trade-off.

Part I

Background

1

Introduction

Pattern recognition is a branch of artificial intelligence concerned with the classification or description of observations. The objective is to classify data based on either a priori knowledge or on statistical information extracted from the data. For that issue, pattern recognition utilities extract previously unknown interesting patterns from data. This extracted knowledge can be used to learn predictive models from a data set of examples that can be used to classify new unseen examples. Essentially, it could be seen as learning patterns from experience in order to predict new information from unseen data.

Different classification problems can be defined depending on the examples available. In this dissertation, we focus on *supervised classification*. Supervised classification assumes the existance of a special variable (the class) whose value we want to predict. Moreover, all the examples of the data set used to learn the classifier are labelled, that is, the value of the class is known.

Classical supervised classification (as defined in the previous paragraph) is focused on the prediction of a single class variable, but many real domains consider more than one class variable, so it would be useful to extend this concept to the prediction of multiple class variables. Recently, this problem has started to be considered by the machine learning community, and has been called *multi-dimensional supervised classification*.

One key aspect of supervised classification is the evaluation of the resultant classifier by means of any score or evaluation function. Usually, the score quantifies the generalization power of a classifier or a classifier induction algorithm. One of the most important scores is the *classification error* (also called *prediction error*). Briefly, the classification error can be seen as the incapability of correctly classifying instances, that is, the probability of misclassification. However, in real-world domains, the true classification error of a classifier is unknown. Moreover, it can not be exactly calculated because the underlying probability distribution is unknown. So, it must be estimated from data. Estimating the classification error of classifiers induced by supervised

learning algorithms is important not only to predict its future error, but also to choose the most promising classifier from a given set (model selection).

This dissertation aims to contribute to the state of the art of both error estimation and multi-dimensional classification fields. Throughout the following chapters, we will explain these contributions.

1.1 Contributions of the dissertation

The contributions presented in this dissertation are two-fold. On the one hand, the contributions related with the error estimation on supervised classification are focused on decomposing the variance of classification error estimators, taking into account its different variance sources. On the other hand, the part related to multi-dimensional classification introduces a multi-objective learning approach for multi-dimensional Bayesian network classifiers. Moreover, we present an application of multi-dimensional Bayesian network classifiers on medical domains.

1.1.1 A general framework for the statistical analysis of the sources of variance of prediction error estimators

The estimation of the classification error is an important task in pattern recognition, not only to predict the future error of a classifier, but also for model selection. In Chapter 3, we present the main aspects of classification error in supervised classification. We show how to estimate the classification error from data, and introduce the most popular error estimators: resubstitution, holdout, repeated holdout, cross-validation, repetition, simple bootstrap and 0.632 bootstrap estimators, with and without stratification.

In Chapter 4, we analyze the statistical properties (bias and variance) of the previously mentioned error estimators for model selection. We present a general framework to analyze the decomposition of the variance of different error estimators considering the nature of the variance (irreducible/reducible variance) and the different sources of sensitivity (internal/external sensitivity).

An extensive empirical study has been performed and, based on the obtained results, we propose the most appropriate error estimations for model selection under different experimental conditions.

1.1.2 Multi-dimensional classification

In Chapter 2 we introduce the supervised classification task and its extension to the prediction of multiple class variables (see Section 2.2.2). This problem is called multi-dimensional classification. We also introduce some classification rules (see Section 5.3) for multi-dimensional probabilistic classifiers which take

into account the multi-dimensional nature of the problem. There are different possibilities in order to adapt the classical one-dimensional classification paradigms to deal with multi-dimensional classification problems. These approaches do not model the real multi-dimensional idiosyncrasy of the problem or generate a huge number of parameters for a high number of class variables. Recently some authors have presented an extension of Bayesian network classifiers to solve multi-dimensional supervised classification problems (van der Gaag and de Waal, 2006; de Waal and van der Gaag, 2007). These models are known as multi-dimensional Bayesian network classifiers (MDBCs).

1.1.2.1 Learning multi-dimensional Bayesian network classifiers by means of a multi-objective approach

In Chapter 6, we present a learning approach of MDBCs following a multi-objective strategy which considers the accuracy of each class variable separately as the functions to optimize.

In order to evaluate the proposed learning approach, we include a study that compares the proposed approach with the main alternatives to deal with multi-dimensional classification. Moreover, this analysis is made for artificial and real-world domains.

1.1.2.2 Application of multi-dimensional Bayesian network classifiers on medical domains

Multiple Sclerosis is a central nervous system disease that is potentially the most common cause of neurological disability in young adults. We find that there are different treatment options avaliable depending on two key aspects (the disease subtype and the expected time to reach a certain severity level), so early identification of these aspects has become highly relevant. Given that we have to predict two correlated class variables: Disease subtype and time to reach certain severity level, we consider the use of multi-dimensional Bayesian network classifiers because they can model and exploit the relations among both variables. Besides, the obtained models can be validated by the physicians by using their expert knowledge due to the interpretability of Bayesian networks.

In Chapter 7 we present an application of MDBCs in order to help a physician to predict the expected progression of the disease and to plan the most suitable treatment. We use the learning approach presented in Chapter 6 adapted to the learning multi-dimensional J/K classifiers instead of unrestricted MDBCs. We test the proposed learning approach by comparing it with the main alternatives to deal with multi-dimensional classification.

1.2 Overview of the dissertation

This dissertation is divided into 8 chapters, which are organized into 5 parts. Part I presents some background concepts and procedures for supervised classification and error estimation. In this part, we also introduce the basic notation used throughout the dissertation. This part is divided into 3 chapters. Chapter 1 is an introduction, with a synthesis of the contributions and how the dissertation is structured. Chapter 2 introduces the basic concepts of supervised classification, with special attention placed on Bayesian network classifiers. In Chapter 3, we introduce the concepts around error estimation and how to estimate the classification error from data.

Part II is devoted to the contributions made in the topic of error estimation. It is only composed of Chapter 4, where we propose a novel framework for the statistical analysis of the sources of variance of classification error estimators.

Part III is focused on multi-dimensional classification. In Chapter 5, we present the basic concepts of multi-dimensional Bayesian classifiers, and in Chapter 6 we present a novel multi-objective learning approach of MDBCs.

In Part IV, composed of Chapter 7, we present an application of multi-dimensional Bayesian network classifier to assist the treatment of multiple sclerosis.

Finally, in Part V (Chapter 8) we summarize all the work of the dissertation, the publications and the future work.

2

Supervised Classification

Supervised Classification (Duda et al., 2001; Bishop, 2006) is one of the main areas in the pattern recognition field. It has been applied in several real-world problems such as bioinformatics, text classification, sentiment analysis or computer vision among others. Informally, it can be seen as learning from experience in order to predict new information from unseen data. Supervised classification assumes the existence of a special variable called the *class variable* or *label* as the value of interest we want to predict given the rest of variables, called *predictive variables* or *features*. For instance we can use genetic information of a patient as features in order to predict the value of any key variable such as the expected progression of a disease. Usually, the given information is represented as a set of labeled examples (called *instances*) of the given concepts, e.g. in the previous example each instance could be composed of the genetic information of a patient and how the disease has evolved in that case.

In this chapter we introduce the general notation used throughout the dissertation and some formal concepts about supervised classification. Finally, we focus on Bayesian network classifiers in supervised classification.

2.1 Notation

A random variable is a function that assigns a numerical value to the outcomes of a random experiment. Let X denote a unidimensional random variable and x a value of the random variable. $\boldsymbol{X} = (X_1, \ldots, X_n)$ represents a n-dimensional random variable where each X_i with $i = 1, \ldots, n$ is a unidimensional random variable. An instance of $\boldsymbol{X}$ is represented by $\boldsymbol{x} = (x_1, \ldots, x_n)$, and $\mathcal{S}_N = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N\}$ denotes a data set of N instances. In general, we use upper-case letters to denote random variables and boldface letters to represent a vector of random variables or instances.

The joint probability distribution of $\boldsymbol{X}$ over a point $\boldsymbol{x}$ is given by $p(\boldsymbol{X} = \boldsymbol{x})$ or just by $p(\boldsymbol{x})$. We denote the marginal probability distribution of $\boldsymbol{X}$ as $p(x)$.

The joint probability distribution of X_i and X_j is represented as $p(x_i, x_j)$ and $p(x_i|x_j)$ represents the conditional probability distribution of X_i given by $X_j = x_j$.

If the random variable has a numerable number of values, we will refer to it as a discrete random variable and, otherwise, if it has a non-numerable number of values, we will refer to it as a continuous random variable.

2.2 Supervised classification

Briefly, supervised classification consists of the learning of a *classification model* or *classifier* from a set of labeled cases or instances (*training set*) by means of a *learning process*. The learnt classifier is a function that assigns a value for the class variable of an unlabeled case described in terms of its features.

In this dissertation, we will focus on *probabilistic classifiers*, that is, classifiers that model the probability distribution of the data and use statistical inference to find the best label for a given instance. These classifiers base the prediction on the estimation of the posterior probability of the class.

2.2.1 Introduction to supervised classification

Formally, the classical approach to *supervised classification* (see Figure 2.1) consists of building a classification model from a data set (training set) in order to predict the value (*label*), $c \in \{1, ..., t\}$ of a class variable C given a set of predictive variables (*feature vector*) $\boldsymbol{X} = (X_1, ..., X_n)$ of an unseen unlabeled instance, $\boldsymbol{x} = (x_1, ..., x_n)$, where $x_i \in \{1, ..., r_i\}$ for $i = 1, ..., n$. We suppose that $(\boldsymbol{X}, C)$ is a random vector with a joint *feature-label probability distribution* $p(\boldsymbol{x}, c)$. Note that in this dissertation we will focus on supervised classification on discrete domains.

A *classifier* ϕ is a function that maps $\boldsymbol{X}$ into C:

$$\begin{aligned} \phi : \Omega_{X_1} \times ... \times \Omega_{X_n} &\to \Omega_C \\ (x_1, ..., x_n) &\mapsto c \end{aligned}$$

where $\Omega_{X_i} = \{1, ..., r_i\}$ and $\Omega_C = \{1, ..., t\}$, and r_i and t are the *cardinalities* of the feature X_i (for $i = 1, \ldots, n$) and the class variable respectively.

A classifier, ϕ, is learned from a training set composed of *independent and identically distributed* (i.i.d.) instances $\mathcal{S}_N = \{(\boldsymbol{x}^1, c^1), ..., (\boldsymbol{x}^N, c^N)\}$ by means of a classifier induction algorithm A, $A(\mathcal{S}_N) = \phi$. We assume that the induction algorithm is a deterministic function of the training set, $\mathcal{S}_N$. Note

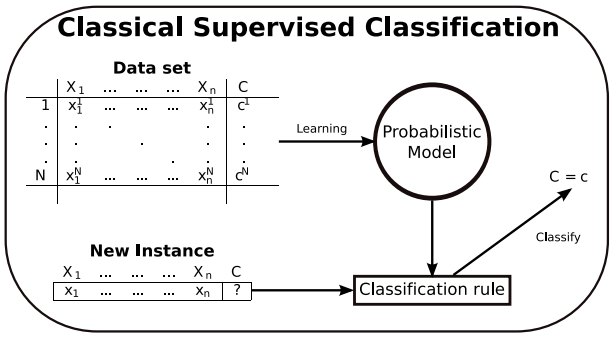


Fig. 2.1. Classical supervised classification based on probabilistic classifiers.

that the training set, $\mathcal{S}_N$, can be considered a random variable which depends on the feature-label random variable $(\boldsymbol{X}, C)$.

A classification problem has an associated misclassification cost function, $loss(c_r, c_e)$, that provides the cost associated to classify a data sample of class c_r in a class c_e. The goal of classifier is to minimize the total misclassification cost. The simplest, but most used, misclassification cost function is called 0/1 *loss function* and is defined as:

$$loss(c_r, c_e) = \begin{cases} 0 \text{ if } c_r = c_e \\ 1 \text{ otherwise} \end{cases}$$

However, for many real-world problems the consequences of different types of mistakes can be dramatically different. For example, let us consider a cancer diagnosis problem. On one hand, if a patient who does not have cancer is incorrectly diagnosed as having cancer, the consequences may be some stress for the patient. On the other hand, if a patient with cancer is incorrectly diagnosed as healthy, the result may be premature death due to the lack of treatment. In this situation, an incorrect diagnosis in a patient with cancer is worse than the opposite, so it must be penalized by the loss function. In Section 2.3 we will tackle the evaluation of the resultant classifiers in depth.

The learning of a supervised classification model can be structured in two different approaches: generative and discriminative (Dawid, 1979; Rubinstein and Hastie, 1997; Jebara, 2004; Santafé et al., 2007).

On the one hand, in a *generative learning* approach the classifier models the underlying joint probability distribution, $p(\boldsymbol{x}, c)$, over all the variables in the problem, included the class. Generative classifiers aim to model the probability distribution that generated the data. Then, the class conditional probability distribution needed to classify new instances can be obtained by means of the Bayes rule:

$$p(c|\boldsymbol{x}) = \frac{p(c,\boldsymbol{x})}{\sum_{c'} p(c',\boldsymbol{x})}$$

Once the classifier is learnt from the training set by means of a classifier induction algorithm, it can be used for predicting the class of a new unlabeled instance by means of any *classification rule*. The optimal classification rule in *probabilistic classification*, called *winner-takes-all*, returns the most likely class value given the features:

$$\hat{c} = argmax_c \{p(c|\boldsymbol{x})\}$$

Examples of probabilistic classifiers are, for example, Bayesian network classifiers (see chapter 2.4).

On the other hand, in a *discriminative learning* approach the classifiers directly model the posterior distribution of the class or just the class boundaries without regard to the underlying class densities. Jebara (2004) introduces the term conditional learning in order to distinguish between discriminative classifiers based on probability distributions that model $p(c|\boldsymbol{x})$, such as logistic regresion (Hosmer and Lemeshow, 1989) or generalized additive models (Hastie and Tibshirani, 1990), and discriminative classifiers that only consider an input-output mapping, such as neural networks (Bishop, 1995) or support vector machines (Vapnik, 1995).

2.2.2 Multi-dimensional supervised classification

Classical supervised classification focuses on the prediction of a single class variable, but many real-world domains consider more than one class variable. So, it would be useful to extend supervised classification to the prediction of multiple class variables. This problem has been called multi-dimensional supervised classification (van der Gaag and de Waal, 2006; de Waal and van der Gaag, 2007; Rodríguez and Lozano, 2010; Bielza et al., 2011).

It should not be confused with other classification tasks such as *multi-class* (Tax and Duin, 2002): problems with a single class variable that can take more than two values, *multi-task* (Caruana, 1997): an inductive transfer approach, where a main task is predicted with the help of the prediction of some extra tasks, or *multi-label* classification (Tsoumakas and Katakis, 2007), where an instance can be classified with a set of labels of different size.

Other classification tasks in pattern recognition can be seen as multi-dimensional classification problems, such as *structured prediction* (Daumé and Marcu, 2005; Bakir et al., 2007), which involves several class variables with a conditional structure among them, or *hierarchical classification* (Dumais and Chen, 2000; Cesa-Bianchi et al., 2006), where there is a hierarchical structure (two or more levels) among the class variables.

Note, however, that multi-label classification can be directly transformed into a multi-dimensional classification problem where each label is represented

as a binary class variable. Then, if a label is assigned to a particular instance in the multi-label classification problem, the value of its equivalent class variable in the multi-dimensional problem is one, and zero otherwise. The opposite, that is, to redefine a multi-dimensional classification task as a multi-label problem is not always straightforward. There are, however, some ways to adapt multi-label techniques to deal with multi-dimensional problems by imposing several restrictions which, from our point of view, are unnatural to the multi-label framework (Ortigosa-Hernández et al., 2012).

Next, we will present an example of a multi-dimensional classification problem and how to transform it into a multi-label problem.

Example: Suppose we consider a multi-dimensional problem where we want to determine the sex, the colour of the eyes and hair colour given several characteristics of a person (as represented in Table 2.1). The problem has five discrete predictive variables and three class variables: Sex, Eyes and Hair. Sex has two possible values: male and female, Eyes has three: blue, dark and green, and Hair has four possible labels: black, blonde, brown and ginger.

inst.	X_1	X_2	X_3	X_4	X_5	Sex	Eyes	Hair	List of labels
(1)	C	C	C	A	A	Male	Blue	Black	{Male, Blue, Black}
(2)	B	A	B	A	D	Male	Dark	Brown	{Male, Dark, Brown}
(3)	A	A	D	A	B	Female	Dark	Blonde	{Female, Dark, Blonde}
(4)	C	B	C	D	A	Male	Green	Blonde	{Male, Green, Blonde}
(5)	B	B	D	A	C	Female	Blue	Ginger	{Female, Blue, Ginger}

Table 2.1. An example of a multi-dimensional problem and the most straightforward way to transfer it to the multi-label domain.

If we wanted to tackle this problem by means of a multi-label algorithm, we would have to force it to fit into the multi-label framework. The most straightforward way to transfer it is to treat each value of each class variable as one independent label, i.e. treat each value Male, Female, Blue, Dark, etc. as a different label. In order to accomplish that, the following conversion has to be carried out:

1. First, define each instance as a list of 3 labels (view the last column of Table 2.1), one per each class variable.
2. Second, deal with the main drawback that this approach has: there are several configuration of labels that are forbidden and/or meaningless in the original multi-dimensional problem. For that reason, several restrictions to the multi-label technique have to be added in order to reflect the true nature of the problem. These constraints are the following:
   a) Fix the number of labels per each instance, e.g. forbid the instances classified as {Male} or {Male, Blue, Black, Ginger}. Each instance must have just 3 labels.

b) Ensure that each instance has just one label per each class variable of the original multi-dimensional problem. We cannot classify an instance as {Male, Female}.

To the best of our knowledge, adapting multi-label techniques by adding several restrictions to deal with this type of problems, as stated in the previous paragraphs, has not been proposed by the research community. Therefore, it is not possible to fit most of the multi-dimensional problems (those that have at least one class variable with more than two values) into the current multi-label framework.

Classical one-dimensional classifiers can also be adapted to deal with multi-dimensional classification problems. One approach consists of constructing a single class variable that models the Cartesian product of all the class variables. The problem arises because this compound class variable can easily end up with an excessively high cardinality. This could lead to computational problems and unreliable parameter estimation because the number of parameters to be estimated in the case of probabilistic classifiers grows exponentially with the number of class variables. Another approach is to develop multiple classifiers, one for each class variable. However, this approach does not model the relationships between the different class variables and, thus, it does not take advantage of the information that they may provide [1].

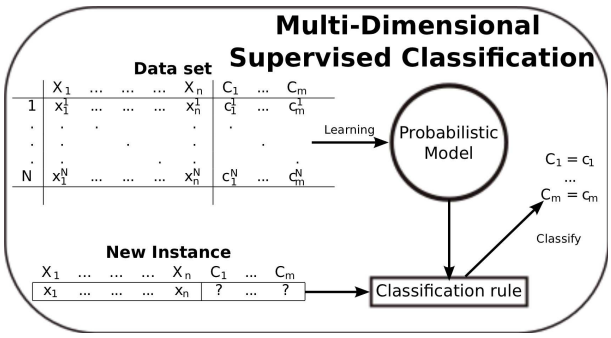


Fig. 2.2. Multi-dimensional supervised classification based on probabilistic classifiers.

Formally, an approach to multi-dimensional supervised classification (see Figure 2.2) consists of building a classifier from training data in order to predict the value of a class vector of m class variables $\boldsymbol{C} = (C_1, \ldots, C_m)$ given the n predictive attributes or features $\boldsymbol{X} = (X_1, \ldots, X_n)$ of an unseen

[1] Note that these drawbacks are closely related with those associated with label power-set and binary relevance approaches to multi-label problems (Read et al., 2011).

unlabeled instance $\boldsymbol{x} = (x_1, \ldots, x_n)$. We suppose that $(\boldsymbol{X}, \boldsymbol{C})$ is a random vector with a joint feature-label probability distribution $p(\boldsymbol{x}, \boldsymbol{c})$.
A multi-dimensional classifier ψ is a function that maps $\boldsymbol{X}$ into $\boldsymbol{C}$:

$$\begin{aligned} \psi : \Omega_{X_1} \times ... \times \Omega_{X_n} &\rightarrow \Omega_{C_1} \times ... \times \Omega_{C_m} \\ (x_1, \ldots, x_n) &\mapsto (c_1, \ldots, c_m) \end{aligned}$$

where $\Omega_{X_i} = \{1, ..., r_i\}$ and $\Omega_{C_j} = \{1, ..., t_j\}$, and r_i and t_j are the cardinalities of the feature X_i (for $i = 1, \ldots, n$) and the class variable C_j (for $j = 1, \ldots, m$) respectively.

A *multi-dimensional classifier* is learnt from a training set $\mathcal{S}_N$, where $\mathcal{S}_N = \{(\boldsymbol{x}^1, \boldsymbol{c}^1), ..., (\boldsymbol{x}^N, \boldsymbol{c}^N)\}$, with a classifier induction algorithm $A(\cdot)$. Given the induction algorithm $A(\cdot)$, the classifier obtained from a training set $\mathcal{S}_N$ is denoted as $\psi = A(\mathcal{S}_N)$. In summary, a multi-dimensional classification problem can be defined as the induction, from a data set $\mathcal{S}_N$, of a classification function ψ that, given a feature vector $\boldsymbol{x}$, returns a class vector $\boldsymbol{c}$.

The winner-takes-all classification rule defined for one-dimension classifiers can be easily generalized to the prediction of more than one class variable: the multi-dimensional classifier returns the most probable combination of class variables given the features. We call it *joint classification rule*:

$$(\hat{c}_1, \ldots, \hat{c}_m) = argmax_{(c_1, \ldots, c_m)}\{p(c_1, \ldots, c_m | x_1, \ldots, x_n)\}$$

However, we can go beyond and develop other classification rules in order to estimate the values of the class variables. In Section 5.3, we propose different classification rules for multi-dimensional probabilistic classifiers.

2.3 Performance evaluation on supervised classification

Once a classifier is constructed, it is necessary to measure its associated error by means of any evaluation score. In this dissertation, we use the *classification error* or *prediction error* of a one-class classifier ψ defined as the probability of wrong classification of unlabeled instances $\boldsymbol{x}$ and it is denoted as $\epsilon(\psi)$:

$$\epsilon(\psi) = p(\psi(\boldsymbol{X}) \neq C) = E_{(\boldsymbol{X}, C)}[1(c, \psi(\boldsymbol{x}))] \tag{2.1}$$

where $1(a, b)$ is a 0/1 loss function whose result is 1 if $a \neq b$ and 0 if $a = b$ (also known as Kronecker delta).

However, in multi-dimensional classification we can measure the correctness of an instance in different ways:

- *Joint evaluation*: This consists of evaluating the estimated values of all class variables simultaneously, that is, it only registers a success if all the

classes are correctly predicted, and otherwise registers an error (see Equation 2.2). This rule generalizes the previous one-class evaluation measure to multi-dimensional classification.

$$\epsilon(\psi) = p(\psi(\boldsymbol{X}) \neq \boldsymbol{C}) = E_{(\boldsymbol{X},\boldsymbol{C})}[1(\boldsymbol{c}, \psi(\boldsymbol{x}))] \tag{2.2}$$

- *Single evaluation*: This consists of checking separately if each class is correctly classified. For example, if we classify an instance $\boldsymbol{x}$ as $(\hat{c}_1 = 0, \hat{c}_2 = 1)$ and the real value is $(c_1 = 0, c_2 = 0)$, we count $\hat{c}_1$ as a success and $\hat{c}_2$ as an error. This approach provides one performance function for each class C_j (for $j = 1, \ldots, m$). The output of this evaluation is a vector $\boldsymbol{\epsilon}$ of size m with the performance function of the multi-dimensional classifier for each class variable:

$$\epsilon_j(\psi) = p(\psi_j(\boldsymbol{X}) \neq C_j) = E_{(\boldsymbol{X},\boldsymbol{C})}[1(c_j, \psi_j(\boldsymbol{x}))] \tag{2.3}$$

  where $\psi_j(\boldsymbol{X})$ is the estimated label of the multi-dimensional classifier for the j-th class variable.

In Chapter 3 we will extend the concepts of evaluation on supervised classification.

2.4 Bayesian network classifiers in supervised classification

Several classification paradigms have been developed to deal with supervised classification problems. In this dissertation we deal with probabilistic classifiers. Specifically, we will focus on Bayesian network classifiers which model the underlying probability distribution by means of a Bayesian network. In this section, we introduce the basic aspects of this classification paradigm.

Bayesian networks (Pearl, 1988) are powerful tools for knowledge representation and inference under uncertainty conditions based on the theory of probabilisic graphical models (PGMs) (Whittaker, 1991; Lauritzen, 1996; Castillo et al., 1997). These formalisms have been extensively used as classifiers (Lauritzen and Spiegelhalter, 1988; Kononenko, 1990; Langley et al., 1992; Dawid, 1992; Ohmann et al., 1996; Mani et al., 1997; McCallum and Nigam, 1998; Friedman et al., 2000; Miyahara and Pazzani, 2000; Hamerly and Elkan, 2001; Movellan et al., 2002; Larrañaga et al., 2005, 2006) in the machine learning field, and have become a classical and well-known classification paradigm.

In this section, first we present the main concepts of graph theory, as the theory of probabilistic graphical models is partly based on this. Then we introduce probabilistic graphical models and finally, we focus on the description of Bayesian network classifiers.

2.4.1 Graph theory

Formally, a graph is a pair $G = (\boldsymbol{X}, \boldsymbol{L})$ where $\boldsymbol{X}$ is a finite set of elements $\{X_1, \ldots, X_n\}$ known as *nodes*, and $\boldsymbol{L}$ is a set of ordered pairs of nodes known as *edges* ($L_{ij} = (X_i, X_j)$). There are two kinds of edges, *directed* and *undirected* edges. We say that an edge L_{ij} is directed when $L_{ij} \in \boldsymbol{L}$ and $L_{ji} \notin \boldsymbol{L}$, and it is represented as $X_i \rightarrow X_j$. We say that an edge L_{ij} is undirected when $L_{ij} \in \boldsymbol{L}$ and $L_{ji} \in \boldsymbol{L}$, and it is represented as X_i–X_j. From now on, the term *arc* will be used to refer to directed edges and the term *edge* will be used only to refer to undirected edges.

A *directed graph* is defined as a graph $G = (\boldsymbol{X}, \boldsymbol{L})$ where every $L_{ij} \in \boldsymbol{L}$ is an arc. An *undirected graph* is defined as a graph $G = (\boldsymbol{X}, \boldsymbol{L})$ where every $L_{ij} \in \boldsymbol{L}$ is an edge. A *Directed Acyclic Graph* (DAG) is a directed graph where there are no *cycles* (a path that starts in a node and ends in the same node).

A *subgraph* of a graph G is a graph whose vertex set is a subset of that of G, and whose adjacency relation is the subset of that of G restricted to this subset. Alternativelly, a supergraph of a graph G is a graph of which G is a subgraph.

Definition 1 *Let $G = (\boldsymbol{X}, \boldsymbol{L})$ be a directed graph and $L_{ij} \in \boldsymbol{L}$ an arc in G. X_i i called* parent *of X_j and X_j is called the* child *of X_i.*

The *d-separation* criterion is a key concept to understand the mechanism behind DAG based PGMs. There are several definitions of d-separation. Here we include the one given by Lauritzen et al. (1990), as it seems to provide the most intuitive interpretation of this concept. The definition is based on the concepts of ancestral set nodes, *moral graph* and the *u-separation* criterion in undirected graphs that we introduce next.

Definition 2 *Let G be a directed graph and X_i and X_j be two nodes of G. X_i is an* ancestral node *of X_j in G if there is a* directed path *from X_i to X_j in G.*

Definition 3 *Let G be a DAG. An* ancestral ordering *of $\boldsymbol{X}$ in G is a total ordering of $\boldsymbol{X}$ where $\forall L_{ij} \in \boldsymbol{L}$, X_i appears before X_j in the order.*

Definition 4 *Let G be a DAG and $\boldsymbol{Y}$ a set of variables of G. The ancestral set of nodes containing $\boldsymbol{Y}$ in G is the set of nodes formed by $\boldsymbol{Y}$ and all the ancestral nodes of the variables contained in $\boldsymbol{Y}$.*

Definition 5 *Let G be a DAG, the* moral graph *associated to G is the graph obtained by adding an arc between parents with a common child and then making all arcs in G undirected.*

Definition 6 *Let G be an undirected graph and let $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$ be three disjoint sets of nodes. We say that $\boldsymbol{W}$ u-separates $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in G if every path in G between a node belonging to $\boldsymbol{Y}$ and a node belonging to $\boldsymbol{Z}$ contains at least one node belonging to $\boldsymbol{W}$.*

The d-separation criterion based on the *u-separation* is defined as:

Definition 7 *Let G be a DAG and let $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$ be three disjoint sets of variables. $\boldsymbol{W}$ d-separates $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in G if $\boldsymbol{W}$ u-separates $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in the moral graph of the smallest ancestral set of nodes which contains $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$.*

Definition 8 *Let G be a bipartite graph (or bigraph) if its vertices can be divided into two disjoint sets $\mathcal{X}$ and $\mathcal{Y}$ such that every edge connects a vertex in $\mathcal{X}$ to one in $\mathcal{Y}$. Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles.*

2.4.2 Probabilistic graphical models

Probabilistic graphical models (PGMs) are mathematical tools used to model joint probability distributions over a set of random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$. PGMs are formed by two basic components, a structure G (a graph that represents the conditional dependence relationships between the random variables) and a set of parameters $\boldsymbol{\Theta}$. Different types of graphs, such as DAGs or undirected graphs, are used to represent the structure of probabilistic graphical models. DAG based PGMs are based on the concept of conditional independence.

Definition 9 *Let $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$ be three disjoint sets of random variables. $\boldsymbol{Y}$ is* conditionally independent *of $\boldsymbol{Z}$ given $\boldsymbol{W}$ if and only if*

$$P(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{w}) = P(\boldsymbol{y}|\boldsymbol{w})$$

for any possible configuration of $\boldsymbol{y}$, $\boldsymbol{z}$ and $\boldsymbol{w}$

The conditional independence is denoted as $CI(\boldsymbol{Y}, \boldsymbol{Z}|\boldsymbol{W})$, and the interpretation is that, given that $\boldsymbol{W}$ is known, the knownledge about $\boldsymbol{Z}$ does not give any extra information about $\boldsymbol{Y}$. In DAG based PGMs, the conditional independence is directly related with the d-separation criterion introduced in the previous section (Pearl, 1988).

Definition 10 *Let $\mathcal{S}$ be a structure of a probabilistic graphical model represented by a directed acyclic graph and let $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$ be three disjoint sets of variables. $\boldsymbol{W}$ d-separates $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in $\mathcal{S}$ if $\boldsymbol{W}$ u-separates $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in the moral graph of the smallest ancestral set of nodes which contains $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$.*

We can factorise the joint probability distribution of $\boldsymbol{X}$ using the *chain rule* as:

$$P(\boldsymbol{x}) = P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|x_1, \ldots, x_{i-1})$$

Without any loss of generality, we can assume that the structure of a DAG based PGM obeys an ancestral ordering such that each node X_i takes the $i-th$ place in the ordering. Given this ancestral ordering, we have that for every ancestral node X_j of X_i, $j < i$.

Taking into account that the set of parents of a node X_i ($\boldsymbol{Pa(X_i)}$) d-separates X_i from any previous node in the ancestral ordering, X_i is conditionally independent of any X_j, $j < i$ given its parents. Therefore, given the structure G the joint probability distribution codified by a DAG based PGM can be factorised as:

$$P(\boldsymbol{x}) = \prod_{i=1}^{n} P(x_i|\boldsymbol{pa(X_i)})$$

Thus, the structure of a PGM indicates how the joint probability can be factorised (reducing the number of parameters needed to model it). Besides the structure, we need a set of parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta} = \{\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_n}\}$ to define the joint probability distribution. Thus, the previous equation should be rewritten as:

$$P(\boldsymbol{x}|\Theta) = \prod_{i=1}^{n} P(x_i|\boldsymbol{pa(X_i)}, \boldsymbol{\theta}_i)$$

2.4.3 Bayesian network classifiers

A Bayesian network classifier is a probabilistic classifier that uses Bayesian networks to model the underlying probability distribution in a particular domain. Based on this model and by means of a classification rule, it predicts the class value of an unlabeled instance.

Formally, a Bayesian network is formed by a pair $B = (G, \boldsymbol{\Theta})$ where G is a directed acyclic graph (DAG) whose vertices correspond to random variables and whose arcs represent conditional (in)dependence relations among variables, and $\boldsymbol{\Theta}$ is a set of parameters. We consider Bayesian networks over a finite set $\boldsymbol{V} = \{X_1, \ldots, X_n, C\}$ where each variable X_i and C takes a finite set of values. $\boldsymbol{\Theta}$ is formed by parameters $\theta_{x_i|\boldsymbol{pa(X_i)}}$ and θ_c for each value that X_i and C can take and for each value assignment $\boldsymbol{pa(X_i)}$ to the set $\boldsymbol{Pa(X_i)}$ of the parents of X_i.

The most common Bayesian network classifier structure is represented as a directed acyclic graph where the class variables are the parents of the rest of the variables, i.e. they are on the top and there are no arcs from predictors to class variables (Friedman et al., 1997). In this case the network B defines a joint probability distribution $p(x_1, \ldots, x_n, c)$ given by:

$$p(x_1, \ldots, x_n, c) = \prod_{i=1}^{n} \theta_{x_i|\boldsymbol{pa(X_i)}} \times \theta_c$$

The Bayesian network can be defined by an expert who is able to list the conditional independences between problem variables or, more frequently, it can be learnt from domain data by means of an induction algorithm. In general, the problem of learning a Bayesian network classifier from data can be seen as: given a training set $\mathcal{S}_N = \{(\boldsymbol{x}^1, c^1), \ldots, (\boldsymbol{x}^N, c^N)\}$ of N instances of $(\boldsymbol{X}, C)$, find a network B that best matches $\mathcal{S}_N$ (Friedman et al., 1997).

There are two main approaches for automatically learning Bayesian networks from data (Neapolitan, 2003): while the first one is based on a score + search process in the space of possible structures, the second tries to detect the conditional independences by means of statistical hypothesis tests.

3

Classification Error

In pattern recognition, the performance evaluation of a classifier or a classifier induction algorithm is a key aspect (Mitchell, 1997). It measures the generalization power of a classifier or a classifier induction algorithm. It can be seen as the capability of correctly classify new instances that have not been used for training the model.

The usual approach to supervised classification consists of learning a classifier from training data by means of a classifier induction algorithm. Each classifier has an associated classification or prediction error (also called *true error*). However, the true error is unknown. So, it must be estimated from data and it is called *estimated classification error*.

Estimating the classification error of classifiers induced by supervised learning algorithms is important not only to predict its future error, but also to choose a classifier from a given set (*model selection*) (Schaffer, 1993). If the goal is to estimate the classification error of a particular classifier, the desired estimator should have low bias and low variance. However, if the goal is the model selection, in order to make fair comparisons, the chosen estimator should have low variance assuming that the bias term is independent of the considered classifier (Kohavi, 1995b).

In this chapter, we introduce the terms of classification error in classical supervised classification (problems with only one class variable) as well as the most popular methods used to estimate the classification error from data.

3.1 Classification Error in Supervised Classification

The *classification error* (prediction error) of a classifier ψ (for 0/1 loss function) is the probability of wrong classification of unlabeled instances $\boldsymbol{x}$ and is defined as follows:

$$\epsilon(\psi) = E_{(\boldsymbol{X},C)}[p(\psi(\boldsymbol{x}) \neq c)] = E_{\boldsymbol{X}}[1 - p(\psi(\boldsymbol{x})|\boldsymbol{x})] \tag{3.1}$$

where $E_{(\boldsymbol{X},C)}$ and $E_{\boldsymbol{X}}$ denote the expected value over the distributions $p(\boldsymbol{x}, c)$ and $p(\boldsymbol{x})$, respectively.

The error of a particular classifier ψ trained in $\mathcal{S}_N$ using the algorithm A is denoted as $\epsilon_{\mathcal{S}_N}(A)$, and it is given by:

$$\epsilon_{\mathcal{S}_N}(A) = \epsilon(\psi = A(\mathcal{S}_N)) \tag{3.2}$$

As S_N is a random variable, $\epsilon_{\mathcal{S}_N}(A)$ can be considered a random variable and we will denote it as ε. The classification error random variable only depends on the data set $\mathcal{S}_N$, so ε can be seen as a function of $\mathcal{S}_N$. The classification error random variable is distributed according to $p(\varepsilon = e) = \sum_{\mathcal{S}_N | \epsilon_{\mathcal{S}_N}(A)=e} p(\mathcal{S}_N)$.

The minimum theoretical classification error is given by the *Bayes classifier* (Koller and Friedman, 2009), ψ_B, which is defined as:

$$\psi_B(\boldsymbol{x}) = \arg_c \max p(c, \boldsymbol{x}) = \arg_c \max\{p(c|\boldsymbol{x})\} = c_B(\boldsymbol{x})$$

We define the **Bayes error** as the classification error of the Bayes classifier:

$$\epsilon(\psi_B) = E_{\boldsymbol{x}}[1 - p(c_B(\boldsymbol{x})|\boldsymbol{x})] = \sum_{\boldsymbol{x}} (1 - p(c_B(\boldsymbol{x})|\boldsymbol{x})) \cdot p(\boldsymbol{x})$$

This value is the highest lower bound of the error of every classifier. It is important to note that Bayes error does not depend on training data or sample size, since the Bayes classifier depends only on the feature-label probability distribution of the domain.

At this point we should clarify that there are two interrelated error estimation problems depending on the purpose of the experimentation: classifier-oriented and algorithm-oriented error estimations.

In **classifier-oriented** error estimation we want to estimate the classification error of a particular classifier $\psi = A(\mathcal{S}_N)$. In applications we are interested in finding the best model to solve a particular task at hand, thereby $\epsilon_{\mathcal{S}_N}(A)$ can be considered an application oriented error estimation. The goal of an application usually consists of selecting the best possible classifier learned using a particular training set $\mathcal{S}_N$. Sometimes, in this context, classification error is also called *conditional error* (Braga-Neto, 2005), in the sense that the error is conditioned to a specific training set, $\mathcal{S}_N$.

In **algorithm-oriented** error estimation, we are not really interested in the performance of a specific classifier induced from a particular training set, $\mathcal{S}_N$. We are concerned with the sensitivity of the learning algorithm to the choice of the training set. So, we will take into account a set of training sets rather than a single one: the set of all training sets of size N. We can define the **expected classification error** as the expectation of the error of a classifier trained with sample sets of size N using the inductor A, and it will be denoted as $\epsilon_N(A)$ or simply as ϵ_N when the inductor algorithm is clear from the context:

$$\begin{aligned}\epsilon_N(A) &= E_{\mathcal{S}_N}[\epsilon_{\mathcal{S}_N}(A)] \\ &= \sum_{\mathcal{S}_N} p(\mathcal{S}_N)\epsilon_{\mathcal{S}_N}(A)\end{aligned}$$

The subscript N is used to differentiate the expected classification error, ϵ_N, from the classification error, $\epsilon_{\mathcal{S}_N}$. Expected classification error is sometimes called *unconditional error* as it does not depend on a particular training set, since it considers all the sets of size N (Braga-Neto, 2005).

It is well known that both errors, $\epsilon_{\mathcal{S}_N}$ and ϵ_N, tend to decrease as the size of the training set N increases. For instance, the consistent classifiers, such as Parzen window or k-nearest neighbor classifiers, converge (under mild conditions) to the Bayes classifier as the number of cases increases to infinity. Besides, the size of the training sets could affect the variance of $\epsilon_{\mathcal{S}_N}$ and ϵ_N. Therefore, N is treated in an explicit way.

Unfortunately, when the feature-label probability distribution $p(\boldsymbol{x}, c)$ is unknown, both $\epsilon_{\mathcal{S}_N}$ and ϵ_N cannot be exactly computed (see Equation 3.1). They have to be estimated from the observed data, $\mathcal{S}_N$, and often, it is crucial to assess the uncertainty attached to this estimation. This uncertainty is related to the variability of the estimation, and it is usually measured in terms of variance. In classifier-oriented experiments, the variance can be used to give a confidence interval on the error associated to the model learned from a given data set $\mathcal{S}_N$, $\epsilon_{\mathcal{S}_N}$. In algorithm-oriented experiments, it is important to give a measure related to the degree of (in)stability of the induction algorithm, A, and to changes in the training set, which can be also measured in terms of variance. The degree of (in)stability of the estimated classification error is, hence, essential to perform comparisons among classifiers and among algorithms.

It should be noted that both estimation problems are closely related and $\epsilon_{\mathcal{S}_N}$ can be used as an estimation of ϵ_N. Moreover, both errors are usually estimated using the same error estimators. For the sake of simplicity, this dissertation focuses on the problem of estimating the error associated to a classifier (classification error or conditional error), $\epsilon_{\mathcal{S}_N}(\psi)$, without loss of generality. From here on, for the sake of brevity, the classification error will be denoted simply as $\epsilon(\psi)$ and the estimated classification error as $\hat{\epsilon}(\psi)$.

3.2 Error Estimation

As we have just mentioned, the feature-label probability distribution $p(\boldsymbol{x}, c)$ is usually unknown, so the error cannot be exactly computed and should be estimated from data. In this section we show how to estimate the classification error of a given classifier and the concepts of bias and variance of the random variables associated to the error estimators.

3.2.1 Classification error estimation

The objective is to estimate the future performance of a classifier induced by a given classifier induction algorithm and a given data set. The estimated classification error can be defined as the capability of correctly classifying instances that have not been used for training the model, so we need a set to train the model (*training set*) and another set to test it (*test set*). For that issue, usually $\mathcal{S}_N$ is partitioned into two parts, one part S is used as the training set and the other part T as the test set.

The estimators of the error analyzed in this dissertation can be thought of as being based on the following estimator of the error:

$$\hat{\epsilon}(A(\mathcal{S}) = \psi; \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x},c)\in\mathcal{T}} 1(\psi(\boldsymbol{x}), c) \tag{3.3}$$

where A is the classifier induction algorithm, $\mathcal{S}$ is the training set, ψ is the classifier inducted from $\mathcal{S}$ using A, $\mathcal{T}$ is the test set, and $1(i, j)$ a 0/1 loss function where $1(i, j) = 1$ iff $i \neq j$ and $1(i, j) = 0$ in any other case (also known as Kronecker delta).

Usually, the error estimator procedure consists of generating (sampling) a set of training-test pair sets $\{(\mathcal{S}^1, \mathcal{T}^1), ..., (\mathcal{S}^k, \mathcal{T}^k)\}$ from a given data set $\mathcal{S}_N$. Then, A classifier is learned from each training set $\mathcal{S}^i$ and, then, the error of the learned classifier is computed in the associated test set $\mathcal{T}^i$, $\hat{\epsilon}_i = \hat{\epsilon}(A(\mathcal{S}^i); \mathcal{T}^i)$. The estimated classification error $\hat{\epsilon}$ is usually given by an appropriate statistic over the computed *intermediate errors* $\{\hat{\epsilon}_1, ..., \hat{\epsilon}_k\}$ for the different training-test pairs, e.g. the average.

3.2.2 Bias and variance

In order to analyze the estimated classification error, it is necessary to consider the concepts of bias and variance of the random variables associated to the error estimators. Let $\hat{\varepsilon}$ be the estimated classification error random variable. The *bias* of an error estimator is defined as the expected estimated classification error value minus the real error value $E[\hat{\varepsilon}] - \varepsilon$. An estimator is said to be *unbiased* if it has zero bias. The *variance* of an error estimator is the variance of the estimation of the error, $E[(E[\hat{\varepsilon}] - \varepsilon)^2]$. For example, in Figure 3.1 we have two estimators, $\hat{\varepsilon}_1$ with a higher bias and lower variance than $\hat{\varepsilon}_1$, and $\hat{\varepsilon}_2$ with lower bias but a higher variance than $\hat{\varepsilon}_1$. Intuitively, the bias measures the average deviation of an estimator, while the variance measures its variability.

An error estimator is said to be *pessimistic* when it has positive bias, i.e. on average it estimates an error higher than the true error, and *optimistic* when it has negative bias.

Clearly, in error estimation we would like an estimator with a bias as low as possible, but this is not necessarily true in model selection. In model selection

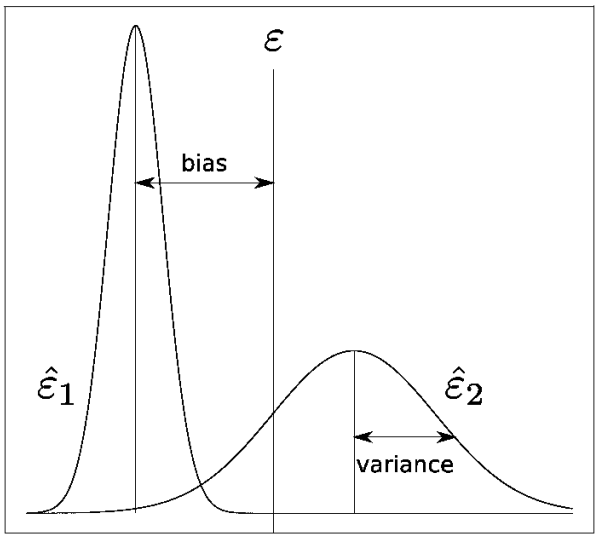


Fig. 3.1. Bias and variance of an estimator

an error estimator with high but equal biases for different induction algorithms is prefereable rather than an error estimator with low but different biases for different induction algorithms. This idea is illustrated in Figure 3.2 by means of an example. The example shows four estimated classification errors obtained using two different estimators, $\hat{\epsilon}$ and $\hat{\epsilon}'$, for naïve Bayes (nB) and C4.5 induction algorithms. In this example, the estimator $\hat{\epsilon}$ is preferable to $\hat{\epsilon}'$, even when it has clearly higher biased estimations. The reason is that $\hat{\epsilon}$ is equally biased with both nB and C4.5 which allows to make fair comparisons among them. Thus, in this dissertation we propose the *absolute values of the bias differences* (ABDs) among different classifiers as an appropriate measure to choose among different error estimators for model selection. In the example of Figure 3.2, the ABD for $\hat{\epsilon}$ among nB and C4.5 is given by $|bias_{nB} - bias_{C4.5}| = 0$ while the ABD for $\hat{\epsilon}'$ is given by $|bias'_{nB} - bias'_{C4.5}|$ which is clearly worse.

As we have just showed, when comparing similar biased classifiers, the variance becomes very important. In Section 4, we will analyze and decompose the variance of error estimators into its sources.

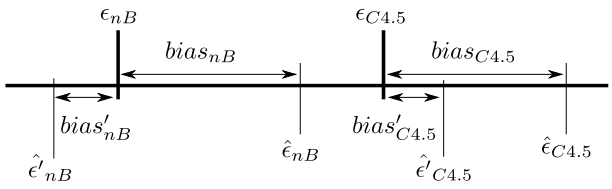


Fig. 3.2. Bias of different error estimators for different classifiers.

3.3 Performance measures

In this section we introduce some metrics used to assess the quality of a classifier. In Section 3.1, we introduced the most commonly used measure, the *classification error*, as the probability that a classifier incorrectly classifies a new instance. We can define the complementary measure, the *accuracy*, as the probability of correctly classifying a new instance:

$$Accuracy(\psi) = E_{(\boldsymbol{X},C)}[p(\psi(\boldsymbol{x}) = c)] = 1 - \epsilon(\psi) \tag{3.4}$$

If we deal with a binary classification problem, that is, a classification problem where the class variable is binary, we can define a confusion matrix and some performance measures based on it. A confusion matrix is a table that shows the performance of a classifier. Each column represents the instances in their predicted class value and each row represents the instances in their real class value (See Table 3.1).

Table 3.1. Confusion Matrix in binary classification problems

		PREDICTED	
		Positive	Negative
REAL	Positive	TP	FN
	Negative	FP	TN

Given an instance, there are four possible values depending on the real and predicted value:

- *True positive (TP)*: This occurs when a positive instance is correctly classified as positive.
- *True negative (TN)*: This occurs when a negative instance is correctly classified as negative.
- *False positive (FP)*: This occurs when a negative instance is incorrectly classified as positive.
- *False negative (FN)*: This occurs when a positive instance is incorrectly classified as negative.

Given a confussion matrix the following scores can be computed:

- *True positive rate (TPR)*, *Recall* or *Sensitivity*:
  $TPR = \frac{TP}{TP+FN}$.
- *False positive rate (FPR)*:
  $FPR = \frac{FP}{TN+FP}$.
- *True negative rate (TNR)* or *Specificity*:
  $TNR = \frac{TN}{TN+FP} = 1 - FPR$.
- *False negative rate (FNR)*:
  $FNR = \frac{FN}{TP+FN} = 1 - TPR$.

Moreover, the previously defined measure, *classification error*, can be defined in terms of the confusion matrix as $\epsilon = \frac{FP+FN}{TP+FN+TN+FP}$.

Usually, in classification problems the classification rules assign the new instance to the class with the highest probability (*winner takes all*), that is, an instance $\boldsymbol{x}$ is positive when $P(1|\boldsymbol{x}) > 0,5$. However, any threshold can be used. Actually, any number between 0 and 1 can be set as a threshold, resulting in different classification functions. There is a graphical representation that allows to evaluate the model considering all the possible thresholds for the conditional probability. It is called *receiver operating characteristic (ROC) curve*, and it is a graphical representation of sensitivity versus 1-specificity (See example in Figure 3.3). It allows to graphically compare different classifiers. The *Area under the ROC curve (AUC)* summarises the curve and gives an idea of the goodness of the model. The closer this value is to 1, the better the model we have.

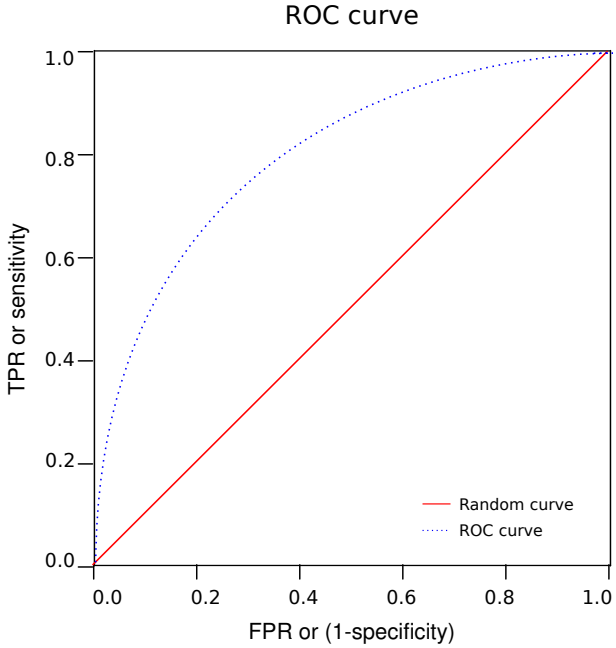


Fig. 3.3. An example of a ROC curve.

The previous measures are oriented to evaluate the performance of classifiers with a single class variable. However, as we show in Section 2.2.2, many real domains consider more than one class variable. This problem is called multi-dimensional supervised classification, and we can define specific performance measures:

- Single accuracy of each class variable by means of a single evaluation (see Section 2.3).
- Mean accuracy (Bielza et al., 2011): This represents the mean of the single accuracy over the m class variables.
- Global accuracy (Bielza et al., 2011): This is the accuracy of the classifier measured by means of a joint evaluation (see Section 2.3) where an instance is correctly classified if and only if it is correctly classified for all the class variables.

3.4 Classification error estimators

In this section, we present the most popular error estimators used by the machine learning community: resubstitution, holdout, repeated holdout, k-fold cross validation, repeated k-fold cross validation, simple bootstrap and 0.632 bootstrap.

3.4.1 Resubstitution

Resubstitution (Devroye and Wagner, 1979) consists of training a classifier using the available data, $\mathcal{S}_N$, and, using Equation 3.3, computing the error with the same data set as the test set. So, the resubstitution error is given by $\hat{\epsilon}_R = \hat{\epsilon}(A(\mathcal{S}_N), \mathcal{S}_N)$ (see Figure 3.4).

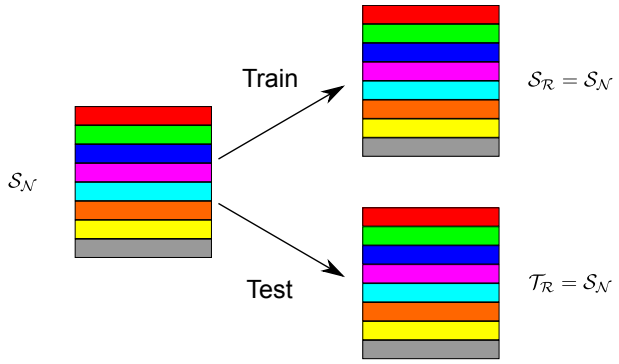


Fig. 3.4. Training-test partition in resubstitution error estimator

This estimator tends to be optimistic due to its strong training-test dependence because $\mathcal{S}_R = \mathcal{T}_R = \mathcal{S}_N$. Clearly, the estimator tends to be more optimistic as the overfitting of a classifier increases. For example, the extreme case of nearest neighbor induction algorithm, A_{NN}, trained in $\mathcal{S}_N$ and tested on the same data obtains zero error, $\hat{\epsilon}(A_{NN}(\mathcal{S}_N), \mathcal{S}_N) = 0$.

3.4.2 Holdout and repeated holdout

Holdout estimator (Larson, 1931; Horst, 1941; McLachlan, 1992) tries to avoid the overfitting problem of resubstitution estimator following a procedure based on a single training-test pair without training-test dependencies. As we show in Figure 3.5, it splits the available data into a disjoint pair training-test, $(\mathcal{S}_H, \mathcal{T}_H)$ where $\mathcal{T}_H = \mathcal{S}_N \setminus \mathcal{S}_H$ and $|\mathcal{S}_H| = \lambda N$ with $0 < \lambda < 1$ (typically, $\lambda = 2/3$). The holdout error is given by $\hat{\epsilon}_H = \hat{\epsilon}(A(\mathcal{S}_H), \mathcal{T}_H)$.

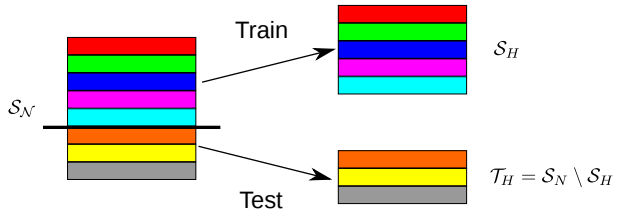


Fig. 3.5. Training-test partition in hold-out error estimator

This estimator avoids the complete training-test dependency of resubstitution estimator by splitting the original data into disjoint training and test sets $\mathcal{S}_H$ and $\mathcal{T}_H$. However, it is known that the error of a classifier tends to decrease as the training set size increases, because more information may be obtained from the data. Thus, the holdout estimator tends to be pessimistic because the size of the training set generated, $|\mathcal{S}_H|$, is smaller than N. On the other hand, holdout tends to have a higher variance than resubstitution because the learned classifier is tested in only $|\mathcal{T}_H| = (1 - lambda)N$ cases, while resubstitution tests the learned classifier in the entire available data $\mathcal{S}_N$.

Repeated holdout estimator is based on iterating a random holdout process k times. The repeated holdout error is computed as the average error among the k holdout estimators, $\hat{\epsilon}_{k\times H} = \frac{1}{k}\sum_{i=1}^{k} \hat{\epsilon}_H^i$.

Repeated holdout estimator reduces the variance of the estimator, maintaining at the same time the negative (pessimistic) bias of the simple holdout estimator. The reduction of the variance is due to splitting the available data into different training-test pairs rather than a single one, reducing the dependence of the estimator from a particular split of the data.

3.4.3 *k*-fold cross validation and repeated *k*-fold cross validation

K-fold cross-validation (k-cv) (Hills, 1966; Anscombe, 1967; Cochran, 1968; Lachenbruch and Mickey, 1968; Stone, 1974; Burman, 1989; Burman et al., 1994; Efron and Tibshirani, 1995; Droge, 1996; Bengio and Grandvalet, 2004; Braga-Neto et al., 2004; Bengio and Grandvalet, 2005) improves repeated holdout estimator by creating more appropriate training-test pairs. The procedure consists of randomly creating a mutually exclusive partition of the data

$\mathcal{P} = \{\mathcal{P}^1, ..., \mathcal{P}^k\}$, where $\mathcal{P}^i \cap \mathcal{P}^j = \emptyset$ for all $i \neq j$, and $\bigcup_{i=1}^{k} \mathcal{P}^i = \mathcal{S}_N$ and each $\mathcal{P}^i$ has equal size when possible $|\mathcal{P}^i| \approx N/k$. Then, the training-test pairs $(\mathcal{S}^i_{kcv}, \mathcal{T}^i_{kcv})$ are created, where $\mathcal{S}^i_{kcv} = \bigcup_{j \neq i} \mathcal{P}^j$ and $\mathcal{T}^i_{kcv} = \mathcal{P}^i$. We consider that these pairs are more appropriate than those used by repeated holdout because (i) all instances in $\mathcal{S}_N$ are used once for testing, and (ii) there is a test-test independency, $\mathcal{T}^i_{kcv} \cap \mathcal{T}^j_{kcv} = \emptyset$ for all $i \neq j$. However, there still is a big dependence between training sets for values of $k > 2$.

The k-cv error is computed as the average error among the k train-test partitions (see Figure 3.6), where $\hat{\epsilon}_i = \hat{\epsilon}(A(\mathcal{S}^i_{kcv}), \mathcal{T}^i_{kcv})$ and the k-cv error is given by $\hat{\epsilon}_{kcv} = \frac{1}{k} \sum_{i=1}^{k} \hat{\epsilon}_i$.

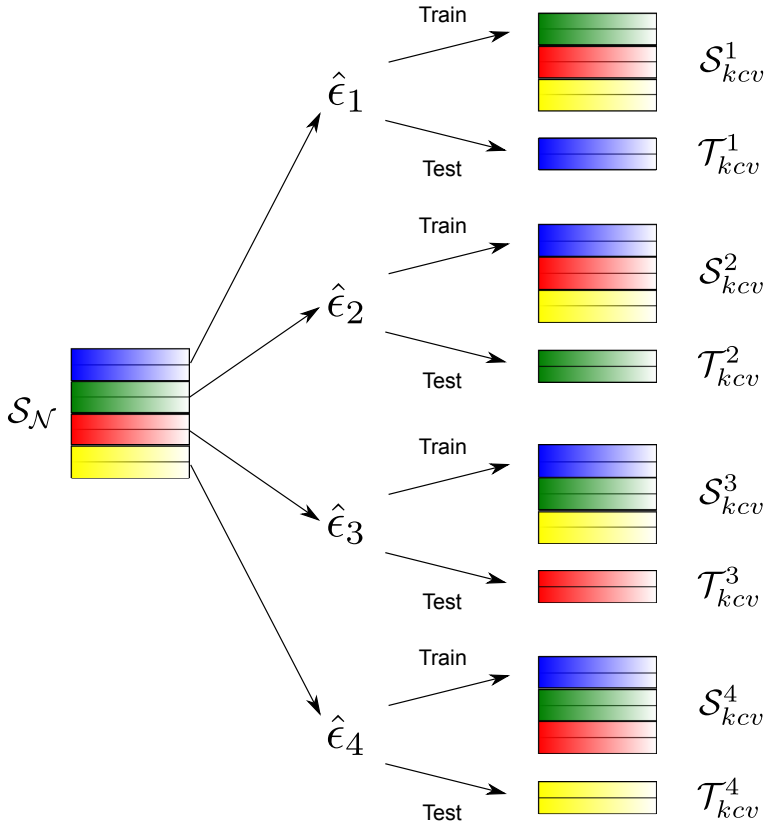


Fig. 3.6. Training-test partition in a 4-fold cross validation error estimator

The most popular k value for k-cv estimator is $k = 10$. However, the selection of k may cause the estimated classification errors to vary dramatically. With $k = 2$ we obtain less dependent intermediate errors $\hat{\epsilon}_i$ because the estimator has test-test and training-training independencies. Note that,

on average, the training sets share $(k-2)/kN$ instances. With $k = N$, the estimator is usually named *leaving-one-out* (LOO) estimator (Mosteller and Tukey, 1968; Fukunaga and Hummels, 1989; Kohavi, 1995a) which is supposed to decrease the variance of k-cv with $k = 10$ at the expense of higher computation requirements, especially when the classifier induction algorithm used is computationally intensive. In general, LOO estimator obtains almost unbiased error estimations, however, it can obtain optimistic estimations in some domains (Lachenbruch and Mickey, 1968).

The k-cv estimator has a negative bias (pessimistic) for the expected classification error on data sets of size N (Braga-Neto et al., 2004) because the intermediate classifiers $\psi_i = A(\mathcal{S}^i_{kcv})$ are learned from training sets $\mathcal{S}^i_{kcv}$ of size $(k-1)N/k$ rather than N. Alternatively, a k-cv error estimator is an unbiased estimator of the expected classification error on data sets of size $N - N/k$, $\epsilon_{N-N/K}$, (Bengio and Grandvalet, 2005; Nadeau and Bengio, 2003).

The repeated approach consists of performing the k-cv error estimator m times for different partitions $\{\mathcal{P}^1, ..., \mathcal{P}^m\}$, where $\mathcal{P}^i = \{\mathcal{P}^i_1, ..., \mathcal{P}^i_k\}$. The estimator constructed by averaging the m different errors estimated by k-cv for different partitions is called *m-repeated k-fold cross-validation* ($m \times k$cv). The error of $m \times k$-cv is computed as the average of the m k-cv errors, $\hat{\epsilon}_{m \times kcv} = \frac{1}{m \cdot k} \sum_{i=1}^{m} \sum_{j=1}^{k} \hat{\epsilon}^i_j$, where $\hat{\epsilon}^i_j = \hat{\epsilon}(A(\mathcal{S}^i_j), \mathcal{T}^i_j)$.

The repeated version of k-cv seems to stabilize the error estimation by reducing the variance (Rodríguez et al., 2010), especially with small data sets (Kohavi, 1995b). However, this reduction is made at the expense of multiplying the computational requirements of k-cv by m, where m is the number of repetitions.

3.4.4 Simple bootstrap and 0.632 bootstrap

The bootstrap estimators (Efron and Tibshirani, 1993) are based on data sets of size N ($\mathcal{S}_B$) obtained by N random samplings with replacement of the original data set $\mathcal{S}_N$. The obtained samples are called the bootstrap samples.

In *simple bootstrap* estimator (see Efron and Tibshirani (1993) Equations 17.17 and 17.22) the k training sets are the bootstrap samples and the test sets are the original data, $\mathcal{T}^i = \mathcal{S}_N$ (see Figure 3.7). The simple boostrap estimation is given by $\hat{\epsilon}_B = \frac{1}{k} \sum_{i=1}^{k} \hat{\epsilon}_i$ where $\hat{\epsilon}_i = \hat{\epsilon}(A(\mathcal{S}^i_B), \mathcal{S}_N)$.

The 0.632 *bootstrap* estimator (06B) (see Efron and Tibshirani (1993) Equation 17.24 and Hastie et al. (2001) Equation 7.54) consists of the resubstitution estimator with a bias correction based on bootstrap samples. Basically, the 0.632 bootstrap estimator appropriately balances the optimistic bias of the resubstitution estimator using a correction based on bootstrap.

The resulting estimator is a weighted average between the resubstitution, and a bootstrap error estimate with similarities to the leaving one out estimate, which is denoted as $\hat{\epsilon}_0$. The 0.632 bootstrap error is given by:

$$\hat{\epsilon}_{632B} = 0.632 \cdot \hat{\epsilon}_0 + 0.368 \cdot \hat{\epsilon}_R \tag{3.5}$$

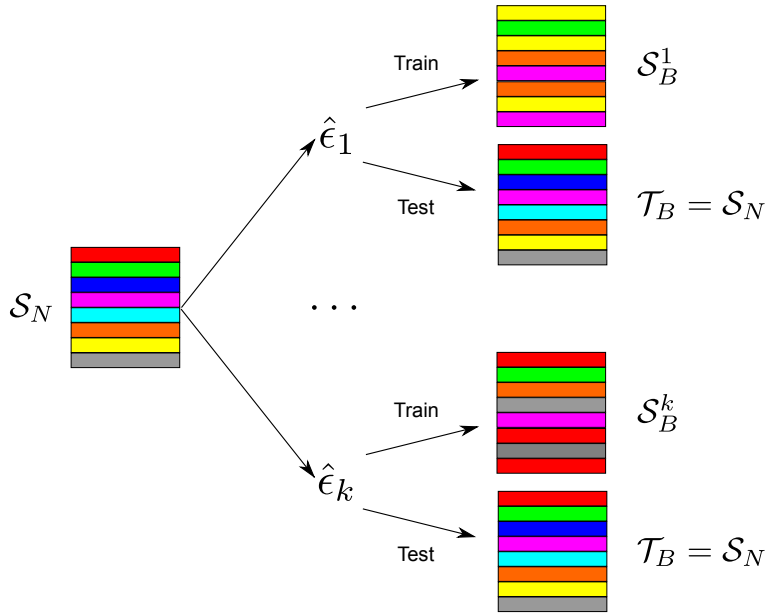


Fig. 3.7. Training-test partition in a simple Bootstrap.

where $\hat{\epsilon}_R = \hat{\epsilon}(A(\mathcal{S}_N), \mathcal{S}_N)$, $\mathcal{T} = \mathcal{S}_N \setminus \bigcup_{i=1}^{k} \mathcal{S}_B^i$ and $\hat{\epsilon}_0$ is given by:

$$\hat{\epsilon}_0 = \frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x},c)\in\mathcal{T}} \frac{1}{|B_{(\boldsymbol{x},c)}|} \sum_{i\in B_{(\boldsymbol{x},c)}} \hat{\epsilon}(A(\mathcal{S}_B^i), \{(\boldsymbol{x},c)\}) \tag{3.6}$$

where $B_{(\boldsymbol{x},c)} = \{i : (\boldsymbol{x},c) \notin \mathcal{S}_B^i\}$ is the set of instances i corresponding to the bootstrap samples $\mathcal{S}^i$ which does not contain the instance $(\boldsymbol{x},c)$ (see Figure 3.8). The k training sets are sampled in the same way as the sets of simple bootstrap. The 0.632 bootstrap estimator has a bias closer to zero than k-cv. This estimator learns the classifiers using training sets $\mathcal{S}^i$ of size $N = |\mathcal{S}_N|$. The 0.632 bootstrap estimator tends to have a variance lower than k-cv (Kohavi, 1995a).

However, the simple and 0.632 bootstrap estimators fail to give the expected result when a classifier is a perfect memorizer due to the high bias of the resubstitution error estimate (Kohavi, 1995a). For example, when the classifier is a nearest neighbor, the estimated error using the resubstitution estimator will be zero and the 0.632 bootstrap error estimator could report a highly biased estimated error $\hat{\epsilon}(\psi) = 0.632 \cdot \hat{\epsilon}_0 + 0.368 \cdot 0 = 0.632 \cdot \hat{\epsilon}_0 \leq 0.632$.

3.4.5 Stratification

Stratification is a procedure that imposes constraints on the sampled data sets, $\{(\mathcal{S}^1, \mathcal{T}^1), ..., (\mathcal{S}^k, \mathcal{T}^k)\}$. The sampled data must contain the same relative frequencies for the class random variable ocurrences that are presented

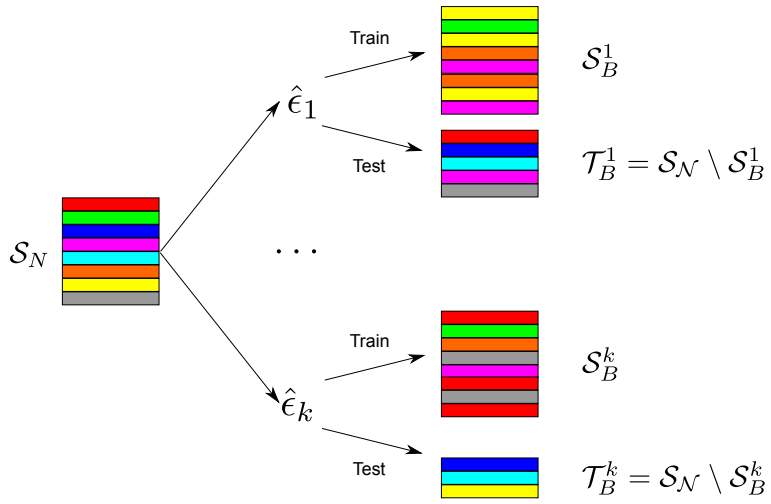


Fig. 3.8. Training-test partition of $\hat{\epsilon}_0$ in a 0.632 Bootstrap.

in the avaliable data. This constraint reduces the variance of the estimations (Breiman et al., 1984) (see Section 4.3 for empirical evidence) since they guarantee that the estimate of the class distribution $p(c)$ from $\mathcal{S}^i$ and $\mathcal{T}^i$ for $i = 1, ..., k$ are equivalent to the estimate obtained from $\mathcal{S}_N$. Intuitively, the estimated marginal distribution of the class $\hat{p}(c)$ may affect the classification of a classifier learned from $\mathcal{S}_N$ and, thus, it can have effects on the estimated classification error. It should be also noted that the Bayes classifier depends on $p(c)$. Since stratification reduces the variance associated to the estimated marginal distribution of the class variable, then, it is expected to reduce the variance of the estimated classification errors. On average, the effect of the reduction of the variance has a higher impact as N decreases. Additionally, the reduction of the variance tends to be greater as the distribution of the class becomes more uniform, because the reduction of the number of possible sets tends to be higher compared to the non-stratified approaches.

Part II

Contributions on Classification Error Estimation

4

A Unified Framework for the Statistical Analysis of the Sources of Variance of Prediction Error Estimators

In the previous chapter, we presented the main aspects of error estimation in supervised classification and its importance not only to predict the future error of a classifier, but also for model selection. The work presented in this chapter is based on Rodríguez et al. (2010) and Rodríguez et al. (2013). Specifically, we analyze the statistical properties: bias and variance, of the most popular error estimators: resubstitution, holdout, repeated holdout, k-fold cross-validation, repeated k-fold cross-validation, simple bootstrap and 0.632 bootstrap estimators, with and without stratification. We present a general framework to analyze the decomposition of the variance of different error estimators considering the nature of the variance (irreducible/reducible variance) and the different sources of sensitivity (internal/external sensitivity).

The general framework for the decomposition of the variance presented in this chapter is used to answer the following interesting questions:

- What part of the variance of an error estimation is independent from the estimator used and what part depends on it?
- What part of the variance is due to sensitivity to changes in the training set and what part is due to sensitivity to changes in the procedure for creating the training-test pairs?

We also present an extensive empirical study and, based on the obtained results, we propose the most appropriate error estimators for model selection under different experimental conditions.

4.1 Introduction

An usual approach to supervised classification consists of the induction of a classifier from training data by means of a classifier learning algorithm. Each classifier has an associated classification error, *true error*. However, the true error is unknown. It can not be calculated because the underlying probability distribution is unknown. So, it must be estimated from data and it is called

estimated error. An estimator of the error of a classifier is a random variable, $\hat{\varepsilon}$, and hence its quality is usually measured by means of its bias and variance. In Section 3.4 we show several estimators of the classification error such as: resubstitution (Devroye and Wagner, 1979), hold-out (Larson, 1931), repeated holdout, k-fold cross-validation (Stone, 1974), repeated k-fold cross validation (Kohavi, 1995a), simple bootstrap and 0.632 bootstrap (Efron and Tibshirani, 1993) estimators.

The mentioned error estimators are based on a data set, $\mathcal{S}_N$, of size N, which usually serves as the training set of the induced classifier. These error estimators are based on a set of training-test data sets generated from the original data set using a sampling procedure. Usually, the creation of the set of training-test pairs is governed by inherent internal factors in the error estimators. Thus, given the estimator, the estimation is completely determined by the classifier induction algorithm, and the generated set of training-test pairs. The analysis consists of a decomposition of the variance into an irreducible part, independent from the estimator used, and a reducible part, which is estimator dependent. Then the reducible part is divided taking into account the two sources of variance: sensitivity to changes in the training set $\mathcal{S}_N$, *external sensitivity*, and sensitivity to changes in the generated set of training-test pairs, *internal sensitivity*. The framework for the decomposition of the variance allows to quantify the contributions of the irreducible variance, the internal sensitivity and the external sensitivity to the total variance for the error estimators considered in this dissertation.

Braga-Neto (2005) proposed an alternative decomposition of the variance of the error estimators into its sources. It divides the whole variance into two terms which are somehow related to the internal and external sensitivities: the internal variance, which measures the variance due to the internal randomness of the error estimator, and the external variance, which measures the variance due to changes in the training set. The main differences of our decomposition are that we explicitly model the internal randomness of the error estimators and, besides, we remove the irreducible variance before decomposing the reducible variance into the internal and external sensitivities.

The error estimators can be used for two different purposes: to obtain a measure of the classification goodness or to select the best classifier among a set of choices. On the one hand, in order to assess a good measure of the quality of a classifier, the bias term is crucial. On the other hand, in model selection, we are quite interested in error estimators with a low variance assuming that the bias term is independent from the considered classifier. Based on the analysis of the variance and the bias among the error estimators considered, in Section 4.3.3 we provide a set of practical suggestions for model selection.

4.2 A general framework for the statistical analysis of the sources of the variance

As we noted in Section 3.2.1, usually an estimation of the error is completely determined by the generated k training-test pairs, $\{(\mathcal{S}^1, \mathcal{T}^1), ..., (\mathcal{S}^k, \mathcal{T}^k)\}$, the classifier induction algorithm, A[1], and the averaging procedure of the intermediate estimated errors determined by the error estimator. Thus, we consider that the variance of the estimation of the error depends on these factors. Moreover, if we fix A and the error estimator, the variance of the estimation uniquely depends on the created training-test sets. In the remainder of this section and for the sake of simplicity, we will consider that the induction algorithm A is fixed.

We assume that the induction algorithm is a deterministic function of the training set, $\mathcal{S}_N$. The process of creating the training-test pairs can be thought of as being determined by k sets of indexes, $\boldsymbol{I} = \{\mathcal{I}_1, ..., \mathcal{I}_k\}$ where $\mathcal{I}_i = \{ind_i^1, ..., ind_i^{|\mathcal{S}^i|}\}$ with $ind_i^j \in \{1, ..., N\}$, assuming that the training set $\mathcal{S}_N$ is ordered. The set of indexes represents the random sampling (with or without replacement) which generates the training test pairs, e.g. in k-cv $\mathcal{S}^i = \{(\boldsymbol{x}^j, c^j) : j \in \mathcal{I}_i\}$ and $\mathcal{T}^i = \mathcal{S}_N \setminus \mathcal{S}^i$ for $i = 1, ..., k$. It should be noted that the sets of indexes represent the internal randomness of the error estimators and this abstraction will enable the exact computation of the part of the variance due to this random component. The training-test pairs sets are completely determined by the set of indexes and the data set. Thus, the sources of variance of the estimators considered in this chapter can be thought of as being determined by the ordered training set $\mathcal{S}_N$ and the set of indexes, $\boldsymbol{I}$, where both random variables are independent given N, $p(\mathcal{S}_N, \boldsymbol{I}) = p(\mathcal{S}_N)p(\boldsymbol{I})$. The distribution of the set of indexes, $p(\boldsymbol{I})$, depends on the random sampling process associated to each error estimator and it will be specified later. The usual process of estimation taking into account the indexes $\boldsymbol{I}$ can be summarized as:

$$\{\mathcal{S}_N, \boldsymbol{I}\} \rightarrow \{(\mathcal{S}^1, \mathcal{T}^1), ..., (\mathcal{S}^k, \mathcal{T}^k)\} \rightarrow \{\hat{\epsilon}_1, ..., \hat{\epsilon}_k\} \rightarrow \hat{\epsilon}(\mathcal{S}_N, \boldsymbol{I}) = \frac{1}{k}\sum_{i=1}^{k} \hat{\epsilon}(\mathcal{S}^i, \mathcal{T}^i) \tag{4.1}$$

where $\hat{\epsilon}(\mathcal{S}^i, \mathcal{T}^i)$ is defined in Equation 3.3.

The estimated error random variable, $\hat{\varepsilon}$, measures the estimated classification error of a classifier induced with A, using a particular error estimation procedure. The estimated error is distributed according to:

$$p(\hat{\varepsilon} = e) = \sum_{\mathcal{S}_N, \boldsymbol{I} | \hat{\epsilon}(\mathcal{S}_N, \boldsymbol{I}) = e} p(\mathcal{S}_N)p(\boldsymbol{I}) \tag{4.2}$$

[1] We assume that the induction algorithm is a deterministic function of the training set, $\mathcal{S}_N$.

The bias random variable, δ, measures the deviation $\epsilon(\mathcal{S}_N) - \hat{\epsilon}(\mathcal{S}_N, \boldsymbol{I})$ (where $\epsilon(\mathcal{S}_N) = \epsilon_{\mathcal{S}_N}(A)$ as defined in equation 3.2) and is distributed according to $p(\delta = e) = \sum_{\mathcal{S}_N, \boldsymbol{I} | \epsilon(\mathcal{S}_N) - \hat{\epsilon}_k(\mathcal{S}_N, \boldsymbol{I}) = \delta} p(\mathcal{S}_N) p(\boldsymbol{I})$.

The estimated error random variable $\hat{\varepsilon}$ can be written as $\hat{\varepsilon} = \varepsilon - \delta$. Thus, its variance can be decomposed into three terms:

$$Var_{(\mathcal{S}_N, \mathbf{I})}[\hat{\varepsilon}] = Var_{\mathcal{S}_N}[\varepsilon] + Var_{(\mathcal{S}_N, \mathbf{I})}[\delta] - 2Cov_{(\mathcal{S}_N, \mathbf{I})}[\varepsilon, \delta] \tag{4.3}$$

If we assume the independence between ε and $\hat{\delta}$, the variance of $\hat{\varepsilon}$ can be decomposed into two terms because $Cov_{(\mathcal{S}_N, \mathbf{I})}[\varepsilon, \delta] \simeq 0$:

$$Var_{(\mathcal{S}_N, \mathbf{I})}[\hat{\varepsilon}] = Var_{\mathcal{S}_N}[\varepsilon] + Var_{(\mathcal{S}_N, \mathbf{I})}[\delta] \tag{4.4}$$

Note that the bias δ depends on the set of indexes $\boldsymbol{I}$, the training set $\mathcal{S}_N$ and on the averaging procedure, rather than on the real error ϵ.

The variance of δ can be decomposed exactly into two terms taking into account its sources of the variance due to the independence of $\boldsymbol{I}$ and $\mathcal{S}_N$ given N:

$$Var_{(\mathcal{S}_N, \boldsymbol{I})}[\delta] = ES + IS \tag{4.5}$$

where ES is the part of the reducible variance due to external factors (changes in the training set, $\mathcal{S}_N$) and IS is the part due to internal factors (changes in the k sets of indexes, $\boldsymbol{I}$). Since the variance can be understood as a sensitivity measure, we refer to ES as *external sensitivity* and IS as *internal sensitivity*.

The definition of both terms is as follows:

$$ES = 1/2(Var_{\mathcal{S}_N}[E_{\boldsymbol{I}}[\delta]] + E_{\boldsymbol{I}}[Var_{\mathcal{S}_N}[\delta]]) \tag{4.6}$$

$$IS = 1/2(Var_{\boldsymbol{I}}[E_{\mathcal{S}_N}[\delta]] + E_{\mathcal{S}_N}[Var_{\boldsymbol{I}}[\delta]]) \tag{4.7}$$

We can demonstrate the exact decomposition of the variance of a random variable, Z, which depends on two random independent variables, X and Y.

Theorem 1 *Given two independent random variables, X and Y, and a third random variable, Z, which depends on X and Y, we have that:*

$$\begin{aligned} Var_{X,Y}[Z] = {} & 1/2(E_X[Var_Y[Z]] + Var_Y[E_X[Z]]) \\ & + 1/2(E_Y[Var_X[Z]] + Var_X[E_Y[Z]]) \end{aligned} \tag{4.8}$$

Proof. By definition of the variance of Z we have that:

$$Var_{X,Y}[Z] = E_{X,Y}[Z^2] - E_{X,Y}[Z]^2 \tag{4.9}$$

We can rewrite this definition by adding and subtracting the term $E_X[E_Y[Z]^2]$ as follows:

$$\begin{aligned} Var_{X,Y}[Z] &= E_{X,Y}[Z^2] - E_X[E_Y[Z]^2] \\ &\quad + E_X[E_Y[Z]^2]) - E_{X,Y}[Z]^2 \\ &= E_X[E_Y[Z^2] - E_Y[Z]^2] \\ &\quad + E_X[E_Y[Z]^2]) - E_X[E_Y[Z]]^2 \\ &= E_X[Var_Y[Z]] + Var_X[E_Y[Z]] \end{aligned} \tag{4.10}$$

Following the same procedure with the term $E_Y[E_X[Z]^2]$ we obtain the following equality:

$$Var_{X,Y}[Z] = E_Y[Var_X[Z]] + Var_Y[E_X[Z]] \tag{4.11}$$

Using Eq. 4.10 and Eq. 4.11 and regrouping the terms we prove the theorem

$$\begin{aligned} Var_{X,Y}[Z] &= 1/2(Var_{X,Y}[Z] + Var_{X,Y}[Z]) \\ &= 1/2(E_Y[Var_X[Z]] + Var_Y[E_X[Z]] \\ &\quad + E_X[Var_Y[Z]] + Var_X[E_Y[Z]]) \\ &= 1/2(E_Y[Var_X[Z]] + Var_X[E_Y[Z]]) \\ &\quad + 1/2(E_X[Var_Y[Z]] + Var_Y[E_X[Z]]) \end{aligned}$$

■

We have decomposed the variance of Z into two additive terms which represent the sources of variance due to variables X and Y respectively (see the two terms of Eq. 4.8). We call $1/2(E_Y[Var_X[Z]] + Var_X[E_Y[Z]])$ the sensitivity of Z with respect to X, and $1/2(E_X[Var_Y[Z]] + Var_Y[E_X[Z]])$ the sensitivity of Z with respect to Y. This property of the variance allows us to decompose the variance of the estimated classification error random variable, $\hat{\epsilon}$, into the sensitivity to changes in the generated set of training-test pairs and the sensitivity to changes in the training set.

We can now study the variance of the estimation as the variance of the real error plus the variance of the deviation of the error. The variance of the real error, $Var_{\mathcal{S}_N}[\varepsilon]$, depends only on the training set used and it is independent of the error estimator used. We call this part of the variance *irreducible variance* because it is common to all the estimators. So, in order to study the inherent properties of the error estimators, it is desirable to subtract it from the total variance, $Var_{(\mathcal{S}_N,\mathbf{I})}[\hat{\varepsilon}] - Var_{\mathcal{S}_N}[\varepsilon] = Var_{(\mathcal{S}_N,\mathbf{I})}[\delta]$. The variance of the bias random variable δ is the variance of the deviation of the estimation. It is the part of the total variance inherent to the estimator used and we call it *reducible variance*. As we noted before, it depends on the ordered training set $\mathcal{S}_N$, the k sets of indexes used, $\boldsymbol{I}$, and on the averaging procedure of the intermediate errors inherent to each error estimator. It should be highlighted that, given $\mathcal{S}_N$, the estimated error is independent of the true error ϵ.

In summary, the decomposition divides the variance into two parts: the irreducible variance, $Var_{\mathcal{S}_N}(\epsilon)$, and the reducible variance, $Var_{(\mathcal{S}_N,\mathbf{I})}(\delta)$. Then,

the reducible variance is decomposed into the external (ES) and internal (IS) sensitivities, respectively. The general decomposition of the variance for the error estimators considered in this dissertation is summarized in Figure 4.1.

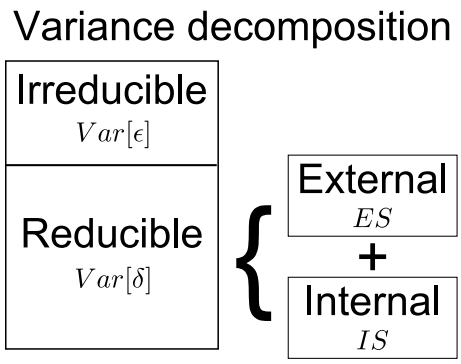


Fig. 4.1. This figure illustrates the general decomposition of the variance into its sources of sensitivity for error estimators.

Next, we will detail the creation of the set of indexes that determine the training-test pairs for the proposed error estimators.

4.2.1 Creation of the set of indexes for the proposed error estimators

In this section, we will detail the process of creating the k training-test pairs for each error estimation method. The training-test pairs are determined by k sets of indexes, $\boldsymbol{I} = \{\mathcal{I}_1, ..., \mathcal{I}_k\}$ where $\mathcal{I}_i = \{ind_i^1, ..., ind_i^{|\mathcal{S}^i|}\}$ with $ind_i^j \in \{1, ..., N\}$, assuming that the training set $\mathcal{S}_N$ is ordered. The usual process of estimation taking into account the indexes $\boldsymbol{I}$ is summarized in Equation 4.1.

4.2.1.1 Resubstitution

In resubstitution all the available data is used to train the model and to test it: $\mathcal{S}_R = \mathcal{S}_N$, $\mathcal{T}_R = \mathcal{S}_N$. So, the set of indexes is given by Equation 4.1 where $k = 1$ and $\mathcal{I}$ is conformed by all the indexes of $\mathcal{S}_N$, $\mathcal{I} = \{ind^1, ..., ind^N\}$:

$$\{\mathcal{S}_N, \boldsymbol{I}\} \rightarrow \{(\mathcal{S}_R, \mathcal{T}_R)\} \rightarrow \hat{\epsilon}_R = \hat{\epsilon}(A(\mathcal{S}_N), \mathcal{S}_N) \tag{4.12}$$

4.2.1.2 Holdout and repeated holdout

In holdout the available data is split into a disjoint training-test pair $(\mathcal{S}_H, \mathcal{T}_H)$ where $\mathcal{T}_H = \mathcal{S}_N \setminus \mathcal{S}_H$ and $|\mathcal{S}_H| = \lambda N$ with $0 < \lambda < 1$. The set of indexes is given by Equation 4.1 where $k = 1$ and $\mathcal{I} = \{ind^1, ..., ind^{\lambda \cdot N}\}$:

$$\{\mathcal{S}_N, \boldsymbol{I}\} \rightarrow \{(\mathcal{S}_H, \mathcal{T}_H)\} \rightarrow \hat{\epsilon}_H = \hat{\epsilon}(A(\mathcal{S}_H), \mathcal{T}_H) \tag{4.13}$$

Repeated holdout repeats k times the holdout process and is summarized by Equation 4.1 with k sets of indexes $\boldsymbol{I} = \{\mathcal{I}^1, \ldots, \mathcal{I}^k\}$ where $\hat{\epsilon}^i_H = \hat{\epsilon}(\mathcal{S}^i_H, \mathcal{T}^i_H)$ and $\mathcal{I}_i = \{ind^1_i, ..., ind^{\lambda \cdot N}_i\}$:

$$\{\mathcal{S}_N, \boldsymbol{I}\} \rightarrow \{(\mathcal{S}^1_H, \mathcal{T}^1_H), \ldots, (\mathcal{S}^k_H, \mathcal{T}^k_H)\} \rightarrow \hat{\epsilon}_{k\times H} = \frac{1}{k}\sum_{i=1}^{k} \hat{\epsilon}^i_H \tag{4.14}$$

4.2.1.3 *k*-fold cross validation and repeated *k*-fold cross validation

In k-fold cross validation the procedure consists of randomly creating a mutually exclusive partition of the data $\mathcal{P} = \{\mathcal{P}^1, ..., \mathcal{P}^k\}$. For each k partition we calculate the error with $\mathcal{T}^i_{kcv} = \mathcal{P}^i$ as the test set and $\mathcal{S}^i_{kcv} = \mathcal{S}_N / \mathcal{T}^i_{kcv}$ as the training set. Each error is computed as $\hat{\epsilon}_i = \hat{\epsilon}(A(\mathcal{S}^i_{kcv}), \mathcal{T}^i_{kcv})$ and the $k$cv error is given by $\hat{\epsilon}_{kcv} = \frac{1}{k}\sum_{i=1}^{k} \hat{\epsilon}_i$.

The set of indexes $\boldsymbol{I}$ is given by Equation 4.1 with k sets of indexes $\boldsymbol{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_k\}$ where each $\mathcal{I}_i = \{ind^1_i, ..., ind^{k/N}_i\}$ is sampled without replacement from $\{1, \ldots, N\}$ and $\mathcal{I}_i \neq \mathcal{I}_j \ \forall i \neq j$:

$$\{\mathcal{S}_N, \boldsymbol{I}\} \rightarrow \{(\mathcal{S}^1_{kcv}, \mathcal{T}^1_{kcv}), \ldots, (\mathcal{S}^k_{kcv}, \mathcal{T}^k_{kcv})\} \rightarrow \hat{\epsilon}_{kcv} = \frac{1}{k}\sum_{i=1}^{k} \hat{\epsilon}_i \tag{4.15}$$

The process of $m \times k$cv is summarized in Equation 4.1 with $m \cdot k$ sets of indexes, $\boldsymbol{I} = \{\boldsymbol{\mathcal{I}^1}, ..., \boldsymbol{\mathcal{I}^m}\}$, where $\boldsymbol{\mathcal{I}^i} = \{\mathcal{I}^i_1, ..., \mathcal{I}^i_k\}$, $\bigcup_{j=1}^{k} \mathcal{I}^i_j = \{1, ..., N\}$ and $\forall_{j \neq j'} I^i_j \bigcap I^i_{j'} = \emptyset$. The estimation is given by $\hat{\epsilon}_{m\times kcv} = \frac{1}{m \cdot k}\sum_{i=1}^{m}\sum_{j=1}^{k} \hat{\epsilon}^i_j$, where $\hat{\epsilon}^i_j = \hat{\epsilon}(A(\mathcal{S}^i_j), \mathcal{T}^i_j)$.

4.2.1.4 Simple bootstrap and 0.632 bootstrap

In Simple bootstrap, the k training sets are the bootstrap samples and the test sets are the original data, $\mathcal{T}^i_B = \mathcal{S}_N$. Each error is computed as $\hat{\epsilon}_i = \hat{\epsilon}(A(\mathcal{S}^i_B), \mathcal{T}^i_B)$ and the simple boostrap estimation is given by $\hat{\epsilon}_B = \frac{1}{k}\sum_{i=1}^{k} \hat{\epsilon}_i$.

The set of indexes $\boldsymbol{I}$ is given by Equation 4.1 with k sets of indexes $\boldsymbol{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_k\}$ where each $\mathcal{I}_i = \{ind^1_i, ..., ind^N_i\}$ is sampled with replacement from $\{1, \ldots, N\}$:

$$\{\mathcal{S}_N, \boldsymbol{I}\} \rightarrow \{(\mathcal{S}^1_B, \mathcal{S}_N), \ldots, (\mathcal{S}^k_B, \mathcal{S}_N)\} \rightarrow \hat{\epsilon}_B = \frac{1}{k}\sum_{i=1}^{k} \hat{\epsilon}_i \tag{4.16}$$

Unfortunately, 06B can not be expressed in terms of Equation 4.1, but the estimation process can be summarized as follows:

$$\begin{aligned} \{\mathcal{S}_N, \boldsymbol{I}\} &\rightarrow \{\mathcal{S}_B^1, ..., \mathcal{S}_B^k, \mathcal{T}\} \rightarrow \{\hat{\epsilon}_0\} \\ \{\mathcal{S}_N\} &\rightarrow \{\hat{\epsilon}_R\} \\ \hat{\epsilon}_{632B} &= 0.632 \cdot \hat{\epsilon}_0 + 0.368 \cdot \hat{\epsilon}_R \end{aligned} \tag{4.17}$$

where $\hat{\epsilon}_R = \hat{\epsilon}(A(\mathcal{S}_N), \mathcal{S}_N)$, $\mathcal{T} = \mathcal{S}_N \setminus \bigcup_{i=1}^{k} \mathcal{S}_B^i$ and $\hat{\epsilon}_0$ is given by:

$$\hat{\epsilon}_0 = \frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x},c)\in\mathcal{T}} \frac{1}{|B_{(\boldsymbol{x},c)}|} \sum_{i\in B_{(\boldsymbol{x},c)}} \hat{\epsilon}(A(\mathcal{S}_B^i), \{(\boldsymbol{x}, c)\}) \tag{4.18}$$

where $B_{(\boldsymbol{x},c)} = \{i : (\boldsymbol{x}, c) \notin \mathcal{S}_B^i\}$ is the set of indexes i corresponding to the bootstrap samples $\mathcal{S}^i$ which do not contain the instance $(\boldsymbol{x}, c)$. The k sets of indexes are equivalent to the sets of simple bootstrap. This estimator learns the classifiers using training sets $\mathcal{S}^i$ of size $N = |\mathcal{S}_N|$. The 0.632 bootstrap estimator tends to have a variance lower than k-cv (Kohavi, 1995a).

4.2.1.5 Stratification

The procedure for generating stratified data sets can be understood as sampling instances from each of the sets $\mathcal{S}_c = \{(\boldsymbol{x}, c) \in \mathcal{S}_N\}$ for $c \in \{1, ..., r\}$ maintaining the relation of instances for each $\mathcal{S}_c$ balanced. Thus the set of indexes can not be considered independent from $\mathcal{S}_N$ given N. However, we consider them independent given N_c for $c \in \{1, ..., r\}$. So, in order to extend the framework to stratified error estimators, we have to consider that $N_c = |S_c|$ (the number of instances of class c) for $c = 1, ..., r$ is (almost) constant for any $\mathcal{S}_N$ obtained from $p(\boldsymbol{x}, c)$. It should be highlighted that for two different sets of size N the probability of having similar N_c values, for $c = 1, ..., r$, tends to one as N tends to infinity.

4.3 Experimentation

In this section we present two empirical studies in order to compare the behavior of the error estimators introduced in Section 3.4. First, we analyze the sources of variances of the estimators considered based on the presented framework for the decomposition of the variance. Then, we analyze the goodness of the estimators in order to make fair comparisons based on the obtained biases and make some practical suggestions.

4.3.1 Experimental setup

In this dissertation, we use data sets obtained from artificial domains because they allow to compute exactly the implied quantities. We have generated random 2-dependences Bayesian classifiers (2-DB) (Sahami, 1996) in order to sample the data sets used in the experimentation. The structure of a 2-DB

classifier allows each predictive variable X_i to have a maximum of 2 dependencies with other predictive variable. The 2-DB classifiers have been randomly generated using the *BNGenerator* software (Ide et al., 2004).

The experimentation has been performed with two different classifiers: naïve Bayes (nB) (Devroye and Wagner, 1979) and C4.5 (Quinlan, 1993). We use these classifiers because they could be thought to have an opposite behavior as regards the sensitivity of the obtained models to changes in the training set: C4.5 induction algorithm is known to suffer from a high sensitivity to changes in the training set, whereas nB is known to be more insensitive because it has a fixed structure and a smaller number of parameters.

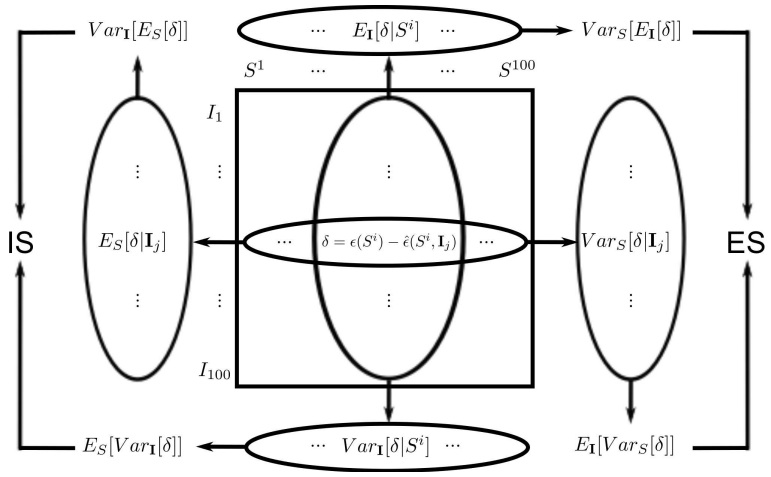


Fig. 4.2. Computation of Internal and External Sensitivities (IS and ES)

In the experimentation we have considered domains with 14 predictive variables plus the class variable. The cardinality of all the variables is 2. We have generated 100 probability distributions represented by different 2-DB classifiers. Then, for each generated probability distribution, we have sampled 100 data sets of 100 instances each. Having fixed a distribution, for each data set $(\mathcal{S}^1, \ldots, \mathcal{S}^{100})$ and each error estimator method, we generate 100 different sets of indexes $\boldsymbol{I}_1, \ldots, \boldsymbol{I}_{100}$ and estimate the error for each classifier (nB and C4.5).

The process performed for the decomposition of the variance of each error estimator following the framework proposed in Section 4.2 is summarized in Figure 4.2. First, we need to calculate the true errors associated to the classifiers learned from each sampled training set, $\{\epsilon(\mathcal{S}^i)\}_i$, and the biases of the

estimated errors associated to each pair of training sets and set of indexes, $\{\delta(\mathcal{S}^i, \boldsymbol{I}_j)\}_{i,j}$ where $\delta(\mathcal{S}^i, \boldsymbol{I}_j) = \hat{\epsilon}(\mathcal{S}^i, \boldsymbol{I}_j) - \epsilon(\mathcal{S}^i)$. Then, for each probability distribution, the decomposition is performed following Equations 4.4 and 4.5. That is, for each data set $\mathcal{S}^i$ we calculate $Var_{\boldsymbol{I}}[\delta|\mathcal{S}^i]$ and $E_{\boldsymbol{I}}[\delta|\mathcal{S}^i]$ and for each set of indexes $\boldsymbol{I}_j$ we calculate $Var_{\mathcal{S}}[\delta|\boldsymbol{I}_j]$ and $E_{\mathcal{S}}[\delta|\boldsymbol{I}_j]$. Finally, we calculate ES and IS as shown in Equations 4.6 and 4.7, respectively. This process is repeated and averaged over the 100 generated probability distributions.
The comparison of the absolute values of the bias differences (ABDs, see section 3.2.2) associated to the estimated errors of nB and C45, $|\delta_{C4.5} - \delta_{nB}|$, is based on the previously computed 10^6 biases (100 distributions $p(\boldsymbol{x}, c) \times 100$ training sets $\mathcal{S}_N \times 100$ set of indexes $\boldsymbol{I}$). The high number of ABDs considered allows to perform robust comparisons using multiple-hypothesis testing procedures, since the p-values tend to decrease with the number of samples.

4.3.2 Experimental results

In this section, we present the experimental results regarding the error estimators considered. The experimental results include the decomposition of the variance of the estimated error, a comparison among the estimators in terms of their variance and a comparison of the estimators in terms of their ABDs. The decomposition of the variance illustrates the different behaviors among the included estimators. The comparisons are performed independently for each family of the error estimators: resubstitution, holdout, k-fold cross-validation and bootstrap families. Then, based on the obtained results the best estimators for each family are selected and we perform a variance and ABD comparison among them. Based on this analysis, in Section 4.3.3 we will propose a set of practical suggestions for the use of error estimators with model selection purposes.

Figures 4.3, 4.6, 4.8, 4.11 and 4.13 show the results of the variance decomposition for the different error estimators (see Figure 4.1). Each bar of the figures represents the total variance of the estimator. The black area of the bar corresponds to the irreducible variance: the variance of the true error ϵ which is shared by all error estimators. The rest of the bar is the reducible variance: the variance of the deviation of the error δ_k which depends on the error estimator used. The white area of the bars is the external sensitivity ES (Eq. 4.6) and corresponds to the part of the variance due to changes in the training set. The grey area is the internal sensitivity IS (Eq. 4.7) and represents the part of the variance due to the internal randomness of the error estimator.

The comparisons among the estimators in terms of variances and ABDs are performed by means of the Friedman test plus Shaffer's static post-hoc test (García and Herrera, 2008). The significance level for all the performed test is 0.01. The samples used for making the comparisons in terms of the variance consist of the estimated 200 variances (100 distributions $p(\boldsymbol{x}, c) \times 2$ classifiers). The estimated variances can be considered paired for each error

estimator since they are based on the same training sets obtained from the same distributions $p(\boldsymbol{x}, c)$. The samples used for making the comparisons in terms of ABD (see Section 3.2.2 for further intuition on this measure) consist of the computed 10^6 ABDs (100 distributions $p(\boldsymbol{x}, c)$ $\times$ 100 training sets $\mathcal{S}_N$ $\times$ 100 set of indexes $\boldsymbol{I}$).

The results of the multiple pairwise comparisons are represented in a critical difference diagram (Demsar, 2006). A critical difference diagram consists of a ranked line. It shows the average rank of the variance (or ABD) of each estimator across all the domains. For example, when an error estimator has an average rank of 2 among 4 error estimators, it means that on average it is the second estimator with the lowest variance (or ABD) across the 100 domains (for instance: 5-fold cross validation in Figure 5.b with a rank of 2.15). Besides, a critical difference diagram includes horizontal lines connecting those estimators without statistically significative differences among them. These estimators belong to the same cluster. For example, if two error estimators are connected by a single horizontal line it means that no statistically significative differences have been obtained in the two pairwise hypothesis testing performed among them (stratified 5-cv and stratified 2-cv in Figure 4.9.b with ranks 2.02 and 2.37 respectively). The obtained clusters represented by the horizontal lines could share some elements. We say that two clusters have statistically significative difference when they do not share any error estimator (there are two clusters {St.B=10, B=10} and {St.B=100, B=1000} in Figure 4.12).

4.3.2.1 Resubstitution

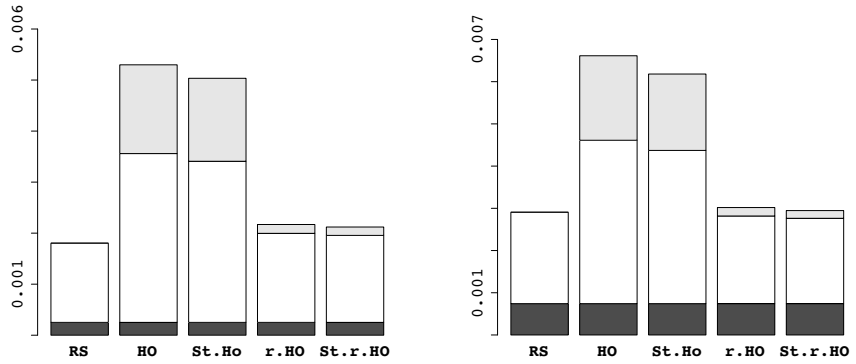
In Figure 4.3, we present the decomposition of the variance of the resubstitution (RS) error estimator (the first bars of Figure 4.3.a and Figure 4.3.b). Note that $IS = 0$ because it does not have internal random procedures. The reducible part of the variance is only formed by the sensitivity to changes in the training set, ES. Clearly, it is the main part of the total variance ($82,5\%$ of the total variance for nB and 65.6% for C4.5).

Due to the absence of internal sensitivity, RS has a low variance even when compared to the most competitive error estimators (see Section 4.3.2.5 for further details). However, the variance is quite different for both classifiers: the variance of C4.5 is 50% bigger than the variance of nB.

Resubstitution is an error estimation method with a bias that strongly depends on the sensitivity of the classifier induction algorithm used and thus it could be inappropriate for making fair comparisons among them (see Section 4.3.2.5 for further details). In our experiments RS shows an average ABD of 0.05.

4.3.2.2 Holdout and repeated holdout

In Figure 4.3, we present the decomposition of the variance of the holdout (HO) and the repeated holdout (r.HO) error estimators in their non-stratified



(a) Naïve Bayes (b) C4.5

Fig. 4.3. Variance decomposition on Resubstitution (RS), Hold Out (HO), Stratified Hold Out (St.HO), repeated Hold out (r.HO) and Stratified repeated Hold out (St.r.HO).

and stratified (St.) versions. In all the cases the most important part of the variance is the external sensitivity ES (60% in the non-repeated version for both classifiers and 80% for nB and 70% for C4.5 in the repeated version). The repeated version also reduces the internal sensitivity to 7% of the total variance in both classifiers.

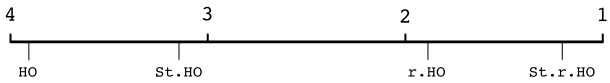


Fig. 4.4. Statistical tests over the variance of holdout error estimators family.

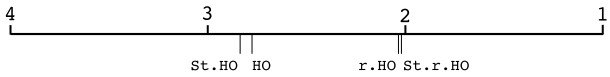


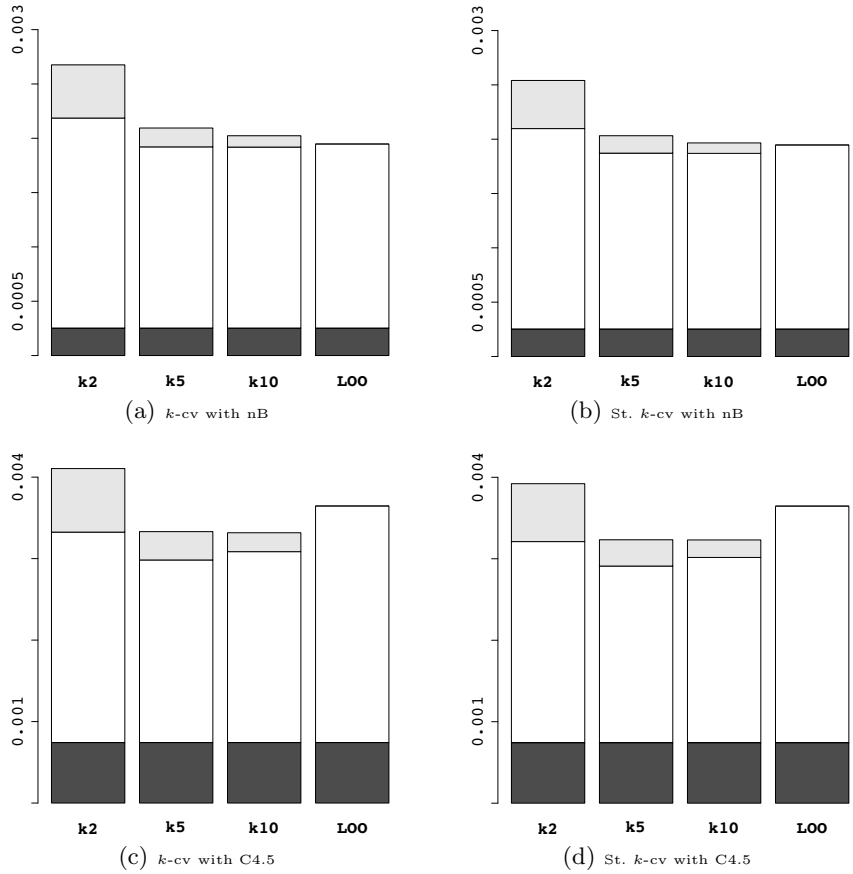
Fig. 4.5. Statistical tests over ABD of holdout error estimators family.

In the repeated version the total variance is drastically reduced to almost half of the variance of the non-repeated version. In general, stratification slighty reduces the total variance, especially for holdout estimator. However, stratification obtains statistically significant better behaviour than the non-stratified versions in terms of variance (see Figure 4.4). The variance of the holdout error estimators for C4.5 is bigger than the variance for nB. Figure 4.4 suggests the use of stratified repeated holdout error estimator in terms of

variance because it shows a statistically significant better behavior compared with the remaining error estimators of the holdout family.

Besides, Figure 4.5 shows that the repeated estimators improve the estimation in terms of ABD with respect to the non-repeated versions, the best being the St.r.HO. It should be noted that St.HO shows worse ABD than HO with statistically significative differences. The average ABD is 0.0081 and 0.0075 for the non-stratified and stratified versions respectively, which is equal for the repeated and non-repeated versions of the holdout estimators.

4.3.2.3 *k*-fold cross-validation and repeated *k*-fold cross-validation



(a) *k*-cv with nB (b) St. *k*-cv with nB

(c) *k*-cv with C4.5 (d) St. *k*-cv with C4.5

Fig. 4.6. Variance decomposition on *k*-fold cross-validation.

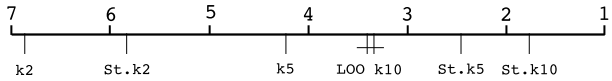


Fig. 4.7. Statistical tests over the variance of k-fold cross-validation.

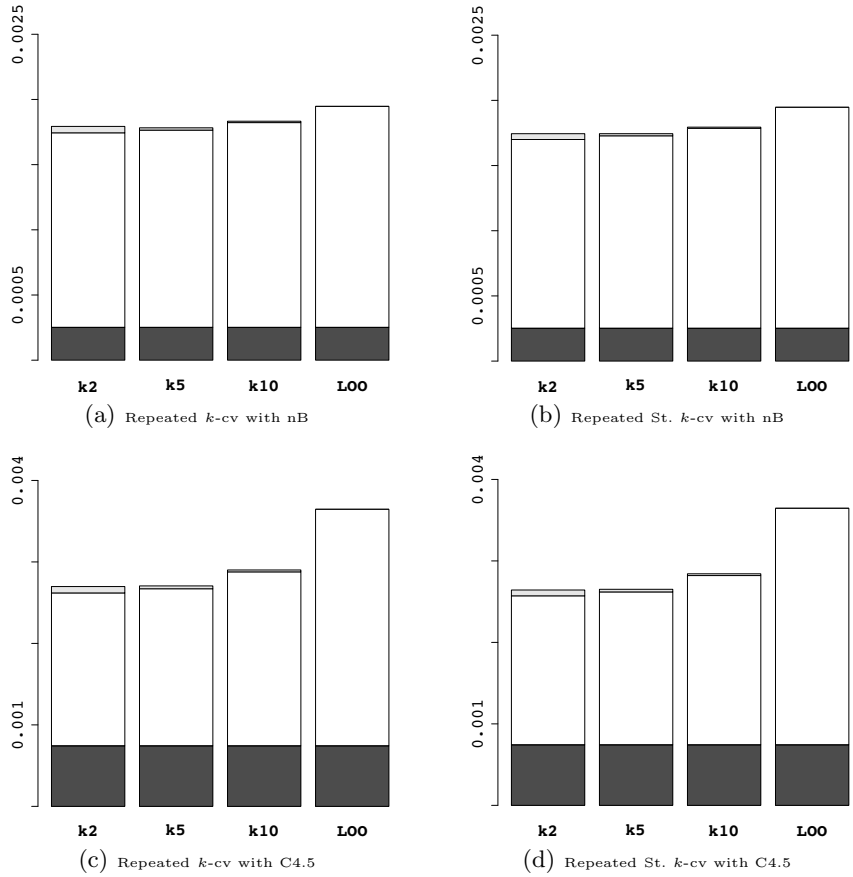


Fig. 4.8. Variance decomposition on repeated k-fold cross-validation

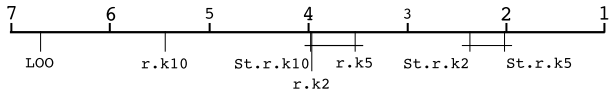


Fig. 4.9. Statistical tests over the variance of repeated k-fold cross-validation.

Figures 4.6 and 4.8 present the decomposition of the variance of k-fold cross-validation (ki for $i = 2, 5, 10$ and LOO for $k = n$) and repeated k-fold cross-validation (r.ki for $i = 2, 5, 10$), respectively. Both figures include the

results of the regular and the stratified (St.) versions of the error estimators. At first sight, we realize that ES dominates the total variance in both error estimators, and it is clearly bigger than IS. It is more than 70% of the total variance for nB and more than 60% for C4.5 in ki . In r.ki, it is more than 80% of the total variance for nB and more than 70% for C4.5. The repeated version reduces both IS and ES, especially IS that becomes a insignificant portion of the whole variance, since its ratio is smaller than 2% of the total variance. Note that $IS = 0$ for LOO because it is a deterministic error estimator given the training set. Stratification slightly reduces both sources of variance, but compared with the repeated procedure the reduction is very small.

Figures 4.7 and 4.9 show a statistically significative improvement in the variance when the stratification is used for all the error estimators of the k-fold cross validation family except LOO. Moreover, Figure 4.10 shows a statistically significative improvement in terms of the ABD for all the error estimators. The best estimators, from the point of view of the ABD are St.r.k5 and St.k5 for the repeated and non-repeated set of estimators respectively. The average ABD obtained in the perfomed experiments is 0.0136, 0.0051 and 0.0025 for k2, k5 and k10, respectively. The stratified and repeated versions slightly reduce the average ABD.

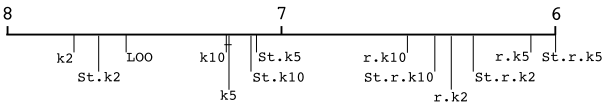
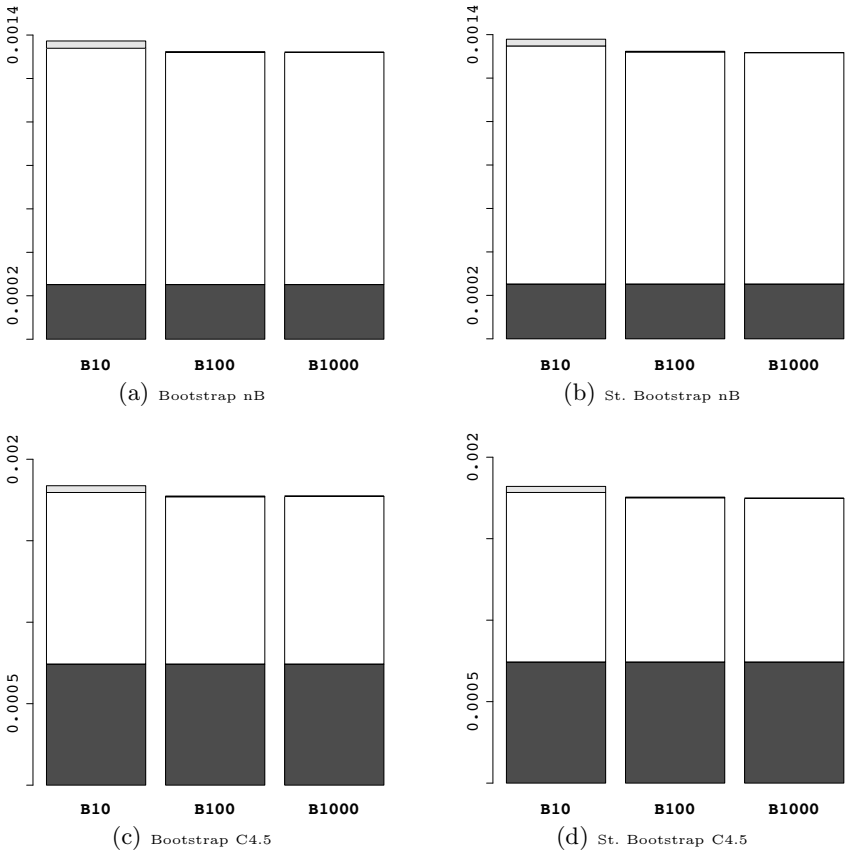


Fig. 4.10. Statistical tests over ABD of k-fold cross-validation error estimators family.

4.3.2.4 Bootstrap and 0.632 bootstrap

Figures 4.11 and 4.13 present the decomposition of the variance with a different number of samples B (10, 100 and 1000) of Simple Bootstrap and 0.632 Bootstrap respectively, in their non-stratified and stratified versions. Similarly to the rest of the error estimators considered in this work, ES dominates the total variance. It is 80% of the total variance for the nB classifier and 60% for C4.5. The internal sensitivity, IS, is extremely small for all the considered bootstrap estimators. For example, it is only 3% of the total variance for $B = 10$.

If we focus on the behavior of the estimators for different B values, we find that the variance of bootstrap error estimators is reduced with statistically significant differences as the value of B increases (see figures 4.12 and 4.14). However, the reduction of the variance from $B = 100$ to $B = 1000$ is very small (a reduction of only 1%). As expected, the variance of bootstrap error estimators for C4.5 is bigger than the variance for nB. Besides, we have found



(a) Bootstrap nB (b) St. Bootstrap nB

(c) Bootstrap C4.5 (d) St. Bootstrap C4.5

Fig. 4.11. Variance decomposition on Simple Bootstrap

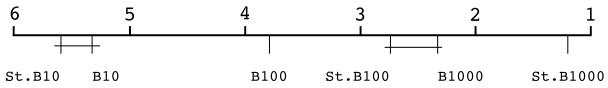
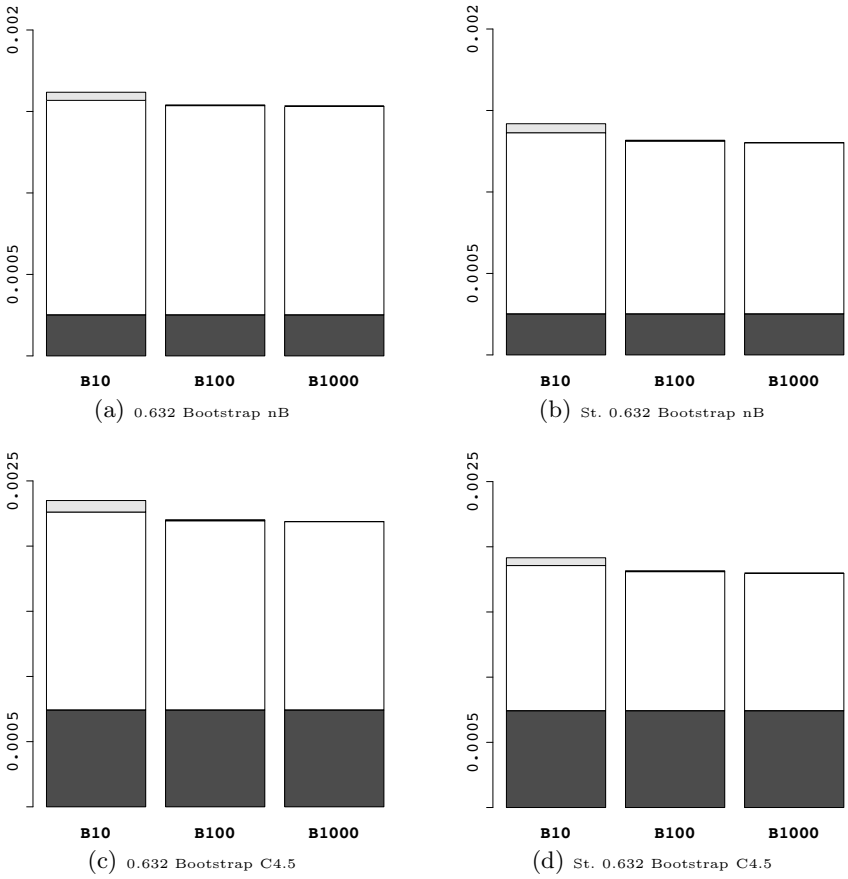


Fig. 4.12. Statistical tests over the variance of simple bootstrap.

that the stratification seems to improve the estimations from the point of view of the variance. However, Figure 4.15 shows that, in terms of ABD, the stratified estimators obtain statistically significative worse results than their non-stratified versions, especially with the 0.632 bootstrap estimators.

All the error estimators of this family have an optimistic bias and they are more biased for C4.5 than nB. The best error estimators from the point of view of the ABD are B1000 and 06B1000 for the simple and 0.632 bootstrap estimators respectively. The computed average ABD is 0.038 for simple bootstrap estimators with and without stratification. On the one hand, in 0.632



(a) 0.632 Bootstrap nB (b) St. 0.632 Bootstrap nB

(c) 0.632 Bootstrap C4.5 (d) St. 0.632 Bootstrap C4.5

Fig. 4.13. Variance decomposition on 0.632 Bootstrap

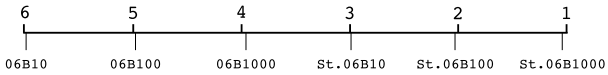


Fig. 4.14. Statistical tests over the variance of 0.632 bootstrap.

bootstrap without stratification the average ABD is 0.011 for $B = 10$ and 0.007 for $B = 1000$. On the other hand, the average ABD is 0.0423 for the stratified versions of 0.632 bootstrap error estimator. It can be concluded that the stratification negatively affects the goodness of 0.632 boostrap estimator for model selection.

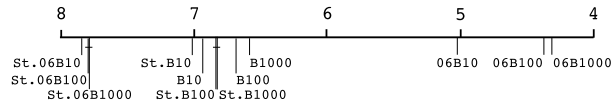


Fig. 4.15. Statistical tests over ABD of bootstrap error estimators family.

4.3.2.5 Error estimators for model selection

In this section we perform a comparison from the point of view of both bias and variance among the set of best candidate error estimators of each family for model selection. The criterion of the selection of the candidates is the behavior in terms of ABDs, because equally biased estimators for different classifiers allows to make fair comparisons among them. The set of selected error estimators are: RS, HO, St.r.HO, St.k5, St.r.k5, B1000 and 06B1000.

Figure 4.16 presents the results of the statistical test among the selected error estimators in terms of variance. HO and B1000 are the error estimators with the highest and the lowest variances respectively. However, there are not statistically significant differences between B1000 and 06B1000. RS shows a notably competitive behavior in terms of variance, since its average rank is 2.29. Unfortunately, RS shows the worst behavior in terms of ABDs (its average ABD is 0.050) and, thus, it seems to be inappropriate for model selection (see Figure 4.17). The best error estimator in terms of ABDs is 06B1000 estimator but it is considerably more computationally intensive than St.r.k5 estimator, which is the second best estimator.

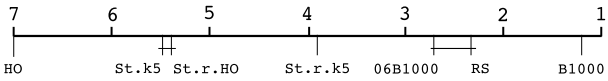


Fig. 4.16. Statistical tests over the variance of the selected error estimators of each family.

The performed decomposition of the variance shows that the main part of the variance of the error estimators considered in this work consists of the external sensitivity. Moreover, some error estimators (bootstrap family and repeated k-fold cross-validation) have reduced the internal sensitivity to a insignificant portion of the whole variance. This suggests that, in the future, in order to design new error estimators we should concentrate on techniques which try to reduce the external sensitivity.

Stratification tends to reduce both the ABD and the variance for most of the error estimators considered. Stratification is particularly effective for the family of k-fold cross-validation error estimators. However, it seems to be inappropriate for model selection for the 0.632 Boostrap error estimators because the stratification seems to considerably increase the average ABD.

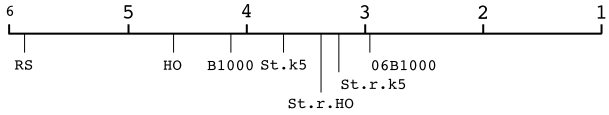


Fig. 4.17. Statistical tests over the ABD of the selected estimators of each family.

4.3.3 Practical suggestions for model selection

In this section we propose a set of error estimators for model selection taking into account the variances and the ABDs computed in the performed experimentation. Taking into account the computational complexity of the implied classifier induction algorithm, we suggest the use of the following error estimators:

- For induction algorithms with low computational complexity, we suggest the use of both 0.632 bootstrap with $B = 1000$ or stratified repeated 5-fold cross-validation estimators.
- For induction algorithms with a high computational complexity, we recommend the use of stratified 5-fold cross validation estimator.

4.4 Conclusions

This chapter analyzes the most popular error estimators from a model selection point of view. The estimators considered are: resubstitution, holdout, repeated holdout, k-fold cross-validation, repeated k-fold cross validation, simple bootstrap and 0.632 bootstrap with and without stratification.

We propose a general framework for the analysis of the sources of variance for the aforementioned estimators. The sources of variance are divided into irreducible and reducible variance, which represent the component independent to the estimator used and the component that depends on the estimator, respectively. Then, the reducible variance is divided into two sources of variability: the external and internal sensitivities. The external sensitivity measures the part of the reducible variance due to changes in the training set, whereas the internal sensitivity measures the part of the reducible variance due to the randomness of the estimator used.

This chapter includes an extensive experimentation in artificial domains for the set of the most popular error estimators, which allows to compute exactly the quantities implied in the variance decomposition. Based on the proposed general framework for the decomposition of the variance, we have empirically studied the relative importance of the different sources of the variances for the error estimators considered. We have observed that in the performed experimentation the external sensitivity is the main part of the variance for all the error estimators considered, whereas for some of the estimators the internal sensitivity is an insignificant portion of the whole variance. This suggests that

in the future we should concentrate our efforts on designing error estimators which try to reduce the external sensitivity.

Besides, we have compared the variance and the Absolute Bias Differences (ABDs) among the different error estimators based on the Friedman plus Shaffer's static hypothesis tests. The comparison in terms of variance allows to select the most stable error estimators whereas the comparison among the bias differences allows to select the most fair error estimator for model selection. Finally, we have proposed a set of practical recommendations in order to select the most appropriate error estimator for making comparisons among classifier induction algorithms. We recommend the use of both 0.632 bootstrap with $B = 1000$ for induction algorithms with low computational complexity or stratified repeated 5-fold cross-validation estimators for induction algorithms with a high computational complexity.

Part III

Contributions on Multi-dimensional Classification

5

Multi-dimensional Bayesian Network Classifiers

In Chapter 2 we introduced the supervised classification task and its extension to the prediction of multiple class variables (see Section 2.2.2). This problem is called multi-dimensional classification. We presented different possibilities in order to adapt the classical one-dimensional classification paradigms to deal with multi-dimensional classification problems. Recently some authors have presented an extension of Bayesian network classifiers to solve multi-dimensional supervised classification problems (van der Gaag and de Waal, 2006; de Waal and van der Gaag, 2007). This models are known as multi-dimensional Bayesian network classifiers (MDBCs).

Van der Gaag (2006) and de Waal and van der Gaag (2007) present for the first time Bayesian network classifiers that use the correlations among class variables to model multi-dimensional classification problems. They propose learning algorithms for a set of structured restricted multi-dimensional Bayesian network classifiers. For these restricted models, the authors prove that the complexity of the learning task is polynomial in the number of predictive variables involved. Recently, other learning approaches have been developed by the community (Borchani et al., 2010; Bielza et al., 2011; Zaragoza et al., 2011b,a). In practical domains multi-dimensional Bayesian networks classifiers have been used for Sentiment Analysis (Ortigosa-Hernández et al., 2012), prediction of HIV protease inhibitors (Borchani et al., 2011) and to assist the treatment of multiple sclerosis (Rodriguez et al., 2012).

In this chapter, we present the basis of multi-dimensional Bayesian network classifiers and the different families of MDBCs developed by the machine learning community. We also revise the different approaches proposed to learn MDBCs from data. Later, in Chapter 6, we will present a novel multi-objective approach to the learning of MDBCs and in Chapter 7 an application on a medical domain.

5.1 Preliminaries

MDBCs generalize the one-class oriented Bayesian network classifiers to domains with more than one class variable. Bayesian networks (Pearl, 1988) have been extensively used as classifiers and have become a classical and well-known classification paradigm. In spite of the popularity of Bayesian network classifiers, few works have taken into account their generalization to multiple class variables (van der Gaag and de Waal, 2006; de Waal and van der Gaag, 2007; Rodríguez and Lozano, 2010; Bielza et al., 2011).

Briefly, a Bayesian network classifier is a probabilistic classifier that uses Bayesian networks to model the underlying probability distribution in a particular domain. Based on this model and by means of a classification rule, it predicts the class value of an unlabeled instance. Usually, the winner-takes-all rule is used because it minimizes ϵ when $\hat{p}(\boldsymbol{x}, \boldsymbol{c})$ is the real underlying distribution.

Next, we will generalize the Bayesian network classifiers definitions introduced in Section 2.4 to the prediction of multiple class variables: Formally, a Bayesian network is formed by a pair $B = (S, \boldsymbol{\Theta})$ where S is a directed acyclic graph (DAG) whose vertices correspond to random variables and whose arcs represent probabilistic relations among variables, and $\boldsymbol{\Theta}$ is a set of parameters. We consider Bayesian networks over a finite set $\mathbf{V} = \{X_1, \ldots, X_n, C_1, \ldots, C_m\}$ where each variable X_i and C_j takes a finite set of values. $\boldsymbol{\Theta}$ is formed by parameters $\theta_{x_i|\boldsymbol{pa}(X_i)}$ and $\theta_{c_j|\boldsymbol{pa}(C_j)}$ for each value that X_i and C_j can take and for each value assignment $\boldsymbol{pa}(X_i)$ and $\boldsymbol{pa}(C_j)$ to the sets $\boldsymbol{Pa}(X_i)$ and $\boldsymbol{Pa}(C_j)$ of the parents of X_i and C_j respectively.

The most common Bayesian network classifier structure is represented as a directed acyclic graph where the class variables are on the top and there are no arcs from predictors to class variables (Friedman et al., 1997). In this case the network B defines a joint probability distribution $p(c_1, \ldots, c_m, x_1, \ldots, x_n)$ given by:

$$p(c_1, \ldots, c_m, x_1, \ldots, x_n) = \prod_{j=1}^{m} \theta_{c_j|\boldsymbol{pa}(C_j)} \prod_{i=1}^{n} \theta_{x_i|\boldsymbol{pa}(X_i)}$$

Next, we will focus on the structure of a MDBC and the possible multi-dimensional classification rules.

5.2 Structure of multi-dimensional Bayesian network classifiers

A MDBC is a generalization of the classical one-class variable Bayesian classifiers for domains with multiple class variables. It models the relationships

between the variables by directed acyclic graphs (DAG) over the class variables and over the feature variables separately, and then connects the two sets of variables by means of a bipartite directed graph. So, the DAG structure $G = (\mathbf{V}, \mathbf{A})$ has the set $\mathbf{V}$ of random variables partitioned into the set $\mathbf{V_C} = \{C_1, \ldots, C_m\}$, $m > 1$, of class variables and the set $\mathbf{V_X} = \{X_1, \ldots, X_n\}$, $n \geq 1$, of feature variables. Moreover, the set of arcs $\mathbf{A}$ can be partitioned into three sets: $\mathbf{A_{CX}}$, $\mathbf{A_C}$ and $\mathbf{A_X}$ with the following properties:

- $\mathbf{A_{CX}} \subseteq \mathbf{V_C} \times \mathbf{V_X}$ is composed of the arcs from the class variables to the feature variables, so we can define the *bridge subgraph* of G as $G_{CX} = (\mathbf{V}, \mathbf{A_{CX}})$.
- $\mathbf{A_C} \subseteq \mathbf{V_C} \times \mathbf{V_C}$ is composed of the arcs between the class variables, so we can define the *class subgraph* of G induced by $\mathbf{V_C}$ as $G_C = (\mathbf{V_C}, \mathbf{A_C})$. It imposes an ancestral order where the classes are before the features.
- $\mathbf{A_X} \subseteq \mathbf{V_X} \times \mathbf{V_X}$ is composed of the arcs between the feature variables, so we can define the *feature subgraph* of G induced by $\mathbf{V_X}$ as $G_X = (\mathbf{V_X}, \mathbf{A_X})$.

In Figure 5.1, we show a MDBC with 3 class variables and 5 features and its partition into the three subgraphs.

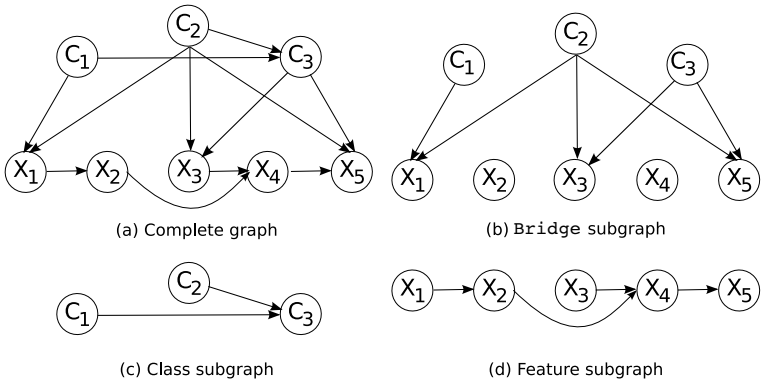


Fig. 5.1. A multi-dimensional class Bayesian network classifier structure and its subgraphs

Depending on the structure of the class subgraph and the feature subgraph, van der Gaag and de Waal (2006) and de Waal and van der Gaag (2007) distinguish the following sub-families of multi-dimensional class Bayesian network classifiers [1]:

[1] In van der Gaag and de Waal (2006) and de Waal and van der Gaag (2007) they use the term *fully* instead of *multi-dimensional* to name the classifiers.

- *Multi-dimensional naïve Bayes classifier (MDnB)*: the class subgraph and the feature subgraph are empty and the bridge subgraph is complete (See Figure 5.2.a).
- *Multi-dimensional tree-augmented classifier (MDTAN)*: both the class subgraph and the feature subgraph are directed trees (See Figure 5.2.b).
- *Multi-dimensional polytree-augmented classifier (MDPoly)*: both the class subgraph and the feature subgraph are polytrees (See Figure 5.2.c).

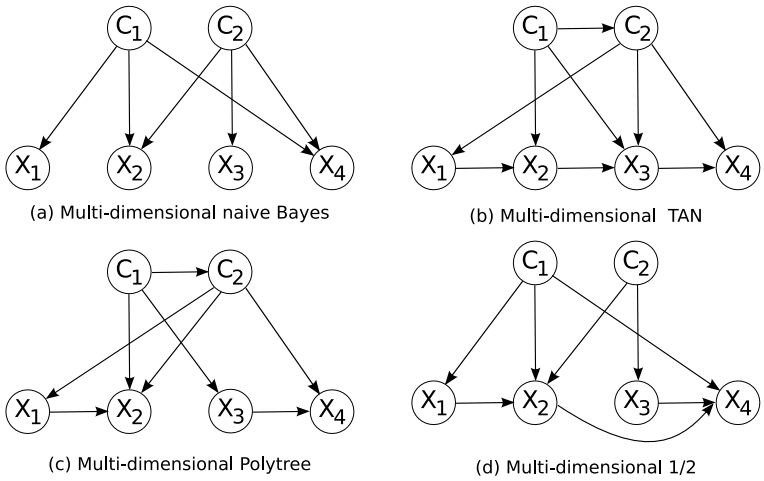


Fig. 5.2. Multi-dimensional class Bayesian classifiers

In addition to these structures, we have considered another structure for experimental purposes:

- *Multi-dimensional J/K dependences Bayesian classifier (MDJ/K)*: This structure allows each class variable C_i to have a maximum of J dependences with other class variables C_j, and each predictive variable X_i to have a maximum of K dependences with other predictive variables X_j (See Figure 5.2.(d)).

In Bielza et al. (2011), the authors have an alternative designation for the different MDBC subfamilies. They denote the MDBCs as *class subgraph structure - feature subgraph structure* MBC. So, they call tree-tree MBC the MDTAN classifier, polytree-polytree MBC the MDPoly classifier and DAG-DAG MBC the MD$J/K$ classifier.

5.3 Multi-dimensional classification rules

In probabilistic classification, the induction algorithm learns a probability distribution $p(\boldsymbol{x}, \boldsymbol{c})$ or $p(\boldsymbol{c}|\boldsymbol{x})$ from the training data and classifies a new instance based on this knowledge. For that purpose, a classification rule must be defined, e.g. winner-takes-all. The multi-dimensional nature of the problem allows us to develop different classification rules that would make no sense in one-class classification because they take into account multiple class variables.

In one-class supervised classification, the most commonly used classification rule returns the most likely class value given the features. We call it *one-dimensional classification rule* or *winner-takes-all*:

$$\hat{c} = arg_c max \{\hat{p}(c|x_1, \ldots, x_n)\}$$

This classification rule can be easily generalized to the prediction of more than one class variable. The multi-dimensional classifier returns the most probable combination of class variables given the features. We call it *joint classification rule*:

$$(\hat{c}_1, \ldots, \hat{c}_m) = arg_{(c_1,\ldots,c_m)} max\{\hat{p}(c_1, \ldots, c_m|x_1, \ldots, x_n)\}$$

However, we can go beyond and develop other classification rules in order to estimate the values of the class variables. We propose another classification rule that consists of marginalizing each class variable for the rest of the class variables simultaneously. We call it *marginal classification rule*. It estimates the value of each class variable C_j (with $j = 1, \ldots, m$) as follows:

$$\begin{aligned} \hat{c}_j &= arg_{c_j} max\{\hat{p}(c_j|x_1, \ldots, x_n)\} \\ &= arg_{c_j} max \left\{ \sum_{\boldsymbol{c}_{\neg j}} \hat{p}(c_j, \boldsymbol{c}_{\neg j}|x_1, \ldots, x_n) \right\} \end{aligned}$$

where $\boldsymbol{c}_{\neg j} = \{c_1, \ldots, c_{j-1}, c_{j+1}, \ldots, c_m\}$.

The previous classification rules estimate the class variables values in one step: the joint classification rule estimates all the class variables simultaneously and the marginal classification rule estimates each class variable separately. However, in multi-dimensional classification we can propose classification rules that estimate the class variables in a procedure with more than one step.

We propose a classification rule based on the marginal values of the class variables. In the first step, each class variable is estimated using the previously defined marginal classification rule. Then, in the final step, we estimate again the value of each class variable using as evidence the class values for the rest of the class variables estimated in the first step. We call it *conditional classification rule*:

$$\widehat{c}_j = arg_{c_j} max\{p(c_j, \widehat{\boldsymbol{c}}^*_{\neg j} | x_1, \ldots, x_n)\}$$

where $\widehat{\boldsymbol{c}}^*_{\neg j}$ are the predicted values for $\{C_1, \ldots, C_{j-1}, C_{j+1}, \ldots, C_m\}$ using the marginal classification rule.

Moreover, we can extend the previous classification rule and continue estimating the class values taking into account the estimated values for the rest of the classes in the previous step. This procedure should continue until a stop criterion is reached, for example when the estimated class values do not change in two consecutive steps. We call it *iterative classification rule*:

$$\begin{aligned} \widehat{c}^0_j = &\ arg_{c_j} max\{p(c_j, \widehat{\boldsymbol{c}}^0_{\neg j} | x_1, \ldots, x_n)\} \\ &\ldots \\ \widehat{c}^s_j = &\ arg_{c_j} max\{p(c_j, \widehat{\boldsymbol{c}}^{(s-1)}_{\neg j} | x_1, \ldots, x_n)\} \end{aligned}$$

where $\widehat{\boldsymbol{c}}^0_j$ are the predicted values for $\{C_1, \ldots, C_{j-1}, C_{j+1}, \ldots, C_m\}$ using the marginal classification rule and $\widehat{\boldsymbol{c}}^{(s-1)}_{\neg j}$ are the predicted values for $\{C_1, \ldots, C_{j-1}, C_{j+1}, \ldots, C_m\}$ in the $s-1$ step using the conditional classification rule.

It must be noted that this criterion does not guarantee convergence, especially for a high number of classes. So, another criterion should be developed, for example a fixed maximum number of iterations.

In Rodríguez and Lozano (2010), a study was carried out comparing the performance in terms of joint and single evaluation of MDBCs using different classification rules. In that work, and for the proposed multi-objective learning, classifiers that use the joint classification rule proved to be slighty more accurate than classifiers using any other classification rule with statistical differences. So, from now on, in this dissertation we will use the joint classification rule.

5.4 Related work

In the last years, several approaches have been proposed in order to learn MDBCs from data. van der Gaag and de Waal (2006) presented a learning algorithm for MDTAN classifiers based on Chow and Liu's algorithms (Chow et al., 1968) to learn the class and feature subgraphs, and searches the bridge subgraph that produces the most accurate classifier. In de Waal and van der Gaag (2007), the same authors presented a partial approach to learn MDPoly classifiers. There is another learning algorithm for MDPoly classifiers proposed by Zaragoza et al. (2011a). They also use Chow and Liu's algorithm (Chow et al., 1968) to learn the class and feature subgraph. In order to learn the

bridge subgraph, they develop a process that adds arcs to the model and searches for the most accurate classifier.

More reciently, Bielza et al. (2011) presented a wrapper learning of what they define as class-bridge decomposable MDBCs. They also presented filter and hybrid learning approaches restricted by an initial ancestral order. The learning of class-bridge decomposable MDBCs is also explored in Borchani et al. (2010) based on a greedy forward selection wrapper approach. Borchani et al. (2011) also presented a learning approach based on the Markov blanket. The authors determine the Markov blanket around each class variable using the HITON algorithm (Aliferis et al., 2003) and use this information to create the MDBC.

Finally, Ortigosa-Hernández et al. (2012) presented a semi-supervised filter learning approach for MDJK classifiers applied to a sentiment analysis problem. This learning approach is based on the mutual information among the variables in the class subgraph and the conditional mutual information among the variables in the feature subgraph. The bridge subgraph is learned adding the arcs between class and feature variables with the highest mutual information.

6

Learning Multi-dimensional Bayesian Network Classifiers by Means of a Multi-objective Approach

In Section 2.2.2 we presented multi-dimensional supervised classification as an extension of the classical supervised classification task to the prediction of multiple class variables. Although there are some methods to adapt classical one-dimensional classifiers to solve multi-dimensional classification problems, van der Gaag and de Waal (2006) and de Waal and van der Gaag (2007) presented Bayesian network classifiers that use the correlations among class variables to model multi-dimensional classification problems. In Chapter 5 we presented the basis of multi-dimensional Bayesian network classifiers and some learning approaches developed by the machine learning community (van der Gaag and de Waal, 2006; de Waal and van der Gaag, 2007; Zaragoza et al., 2011a; Bielza et al., 2011; Borchani et al., 2010; Ortigosa-Hernández et al., 2012).

In this chapter, we deal with the problem of learning Bayesian network classifiers for multi-dimensional supervised classification problems. We present a learning approach following a multi-objective strategy which considers the single classification accuracy of each class variable as the functions to optimize. The solution of the learning approach is a Pareto set of non-dominated multi-dimensional Bayesian network classifiers and their accuracies for the different class variables, so a decision maker can easily choose by hand the classifier that best suits the particular problem and domain. The proposed learning approach does not impose any restrictions in the learnt structure as it is defined in 5.2, thus relaxing the constraints of previously learning approaches.

6.1 Introduction

As we show in the previous chapter, recent works have developed single-objective learning algorithms for restricted multi-dimensional Bayesian network classifier structures (MDTAN (de Waal and van der Gaag, 2007), MD-Poly (Zaragoza et al., 2011a), MD$J/K$ (Ortigosa-Hernández et al., 2012) and

CB-Decomposable multi-dimensional Bayesian network classifiers (Borchani et al., 2010)). Bielza et al. (2011) presented a wrapper learning of unrestricted multi-dimensional Bayesian network classifiers and a filter and hybrid learning approaches restricted by an initial ancestral order. Finally, Borchani et al. (2011) presented a learning approach based on Markov Blankets. In this chapter, we present a multi-objective learning approach of multi-dimensional Bayesian network classifiers with no structural restrictions given the definition presented in Section 5.2. Although there are some supervised classification approaches by means of multi-objective optimization (Handl et al., 2007; Jin and Sendhoff, 2008), as far as we know this is the first one that deals with multi-dimensional classification.

The use of a multi-objective learning approach returns a Pareto set of classifiers with different accuracy values for each of the classes. There are mainly two reasons to return a set of classifiers instead of returning only one classifier:

- Expert validation and selection: The interpretability of the multi-dimensional Bayesian network classifiers makes it possible to study the relationships among all the variables involved in the problem and lets the expert check whether or not they are correct (Lacave et al., 2007). The expert can then validate the classifiers using his/her expert knowledge, and select the most suitable in terms of the modeled relations.
- Selection based on the accuracy: An expert can select the most suitable classifier, giving relative importance to the accuracy of each class variable, using the mean accuracy or the global accuracy as the selection key (these evaluation measures are explained in Section 6.3.1.1). The expert can even select the best classifier for each class variable in order to classify a new instance.

This chapter is organized as follows: First, we present the proposed learning approach by briefly introducing the concept of multi-objective optimization and the selected multi-objective algorithm for the learning. Then, we present the adaptation made in the algorithm in order to deal with a multi-dimensional learning problem. Finally, we present the performed experimentation and the results.

6.2 Multi-objective learning approach

In this section we present the proposed learning approach. We set out the learning of the structure S of a multi-dimensional Bayesian network classifier B as a multi-objective optimization problem where we try to maximize the accuracy of each class variable. The search space is composed of all the possible structures of multi-dimensional Bayesian network classifiers. The objective functions of the multi-objective optimization problem are the single

accuracies of the classifiers for each class variable separately.
Ideally, we would like to exactly calculate the accuracy of a classifier, but in most real world problems the feature-label probability distribution $p(\boldsymbol{x}, \boldsymbol{c})$ is unknown. So, the accuracy is also unknown, it cannot be exactly computed, and thus, must be estimated from data. For that issue, we use an estimation of the acccuracy as the objective functions for the multi-objective optimization problem. In Section 3.4 we presented the most popular error estimators used in supervised classification, we have chosen k-fold cross-validation as the objective functions.

6.2.1 Multi-objective optimization

A *multi-objective optimization problem* (MOP) is an optimization problem with multiple objectives measured with different performance functions, usually in conflict with each other, and a set of restrictions. The optimization consists of finding a particular solution which gives the values of all the objective functions acceptable to a decision maker (Osyczka, 1985), who has to choose the preferred optimal solution. The aim is to find good compromises (trade-offs) rather than a single solution (Coello et al., 2006).

An important concept in MOP is the *Pareto dominance*: A vector $\boldsymbol{u} = (u_1, \ldots, u_m)$ is said to *dominate* $\boldsymbol{v} = (v_1, \ldots, v_m)$ on maximization (denoted by $\boldsymbol{u} \succeq \boldsymbol{v}$) if and only if $\boldsymbol{u}$ is partially more than $\boldsymbol{v}$. That is $\forall i \in \{1, \ldots, m\}, u_i \geq v_i \wedge \exists j \in \{1, \ldots, m\} : u_j > v_j$.

The solution of a multi-objective optimization problem is a Pareto set composed of solutions non-dominated among them (Stadler, 1988). When plotted in a objective function space, the non-dominated solutions form a Pareto front. In our case, each point in the Pareto front represents the accuracy for each class variable of a given multi-dimensional classifier (see Figure 6.1).

In this work, we use a well known multi-objective evolutionary algorithm: the *Multi-objective evolutionary algorithm based on decomposition* (Zhang and Li, 2007; Li and Zhang, 2009) (MOEA/D).

MOEA/D explicity decomposes the multi-objective optimization problem into a number of scalar optimization subproblems and optimizes them simultaneously. The problem is decomposed using a Tchebycheff approach (Miettinen, 1999) that generates P different single-objective subproblems using different weight vectors $(\boldsymbol{\lambda^1}, \ldots, \boldsymbol{\lambda^P})$. Each weight vector $\boldsymbol{\lambda^j} = (\lambda_1^j, \ldots, \lambda_m^j)$ assigns different weights to the m objectives, which defines a single-objective perfomance function for each particular decomposition of the problem. MOEA/D solves these subproblems by evolving a population composed of the best solutions found so far for each subproblem. At each step of the algorithm, and for each subproblem j, it develops a new solution by crossing solutions in the neighbour of the j-th subproblem (the neighborhood of each weight vector $\boldsymbol{\lambda^j}$ is defined as a set of its T closest weight vectors) and replaces a maximum of

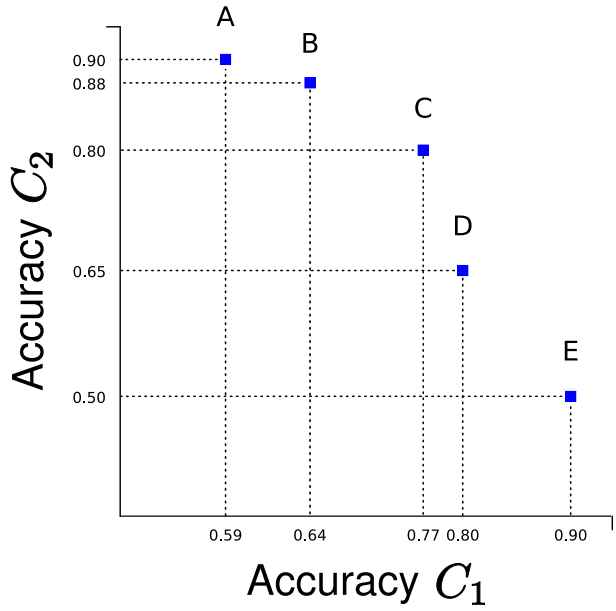


Fig. 6.1. The solution of a multi-objective optimization learning approach. A, B, C, D and E are non-dominated multi-dimensional classifiers.

nr (number of replacements) solutions in the neighborhood that are improved by the new solution. Finally, in order to return a Pareto set, it maintains an external population with the best non-dominated solutions found so far.

6.2.2 MOEA/D adaptation to learning

In order to avoid overfitting and obtain a fair accuracy estimation, the data set used to train the classifier must be independent of the data set used to test it (Braga-Neto, 2005). To that end, we have made a slight modification to the original MOEA/D algorithm to guarantee this independence assumption (see Figure 6.2).

The basic idea is to split the data set into two different parts: $\mathcal{S}^1$ with 3/4 of the instances of $\mathcal{S}_N$, and $\mathcal{S}^2$ with the rest of the instances. First, we perform the structural learning of the multi-dimensional Bayesian classifiers. The data set $\mathcal{S}^1$ is used to guide the multi-objective learning process by means of a repeated stratified k-cv procedure ($r \times k$cv) (step 2.2 on Figure 6.2). We have selected $r \times k$cv estimator since it has shown competitive behaviours for model selection (see Chapter 4). So, the objective functions of the multi-objective problem are:

$$\boldsymbol{r \times kcv(\psi)} = [r \times kcv_1(\psi), \ldots, r \times kcv_m(\psi)]$$

where $\boldsymbol{\psi}$ is a multi-dimensional classifier.

- **Step 1: Initialize.**
  - **Step 1.1**: Compute Euclidean distances between the P weight vectors and find the T closest neighbors of each subproblem.
  - **Step 1.2**: Generate an initial population.
  - **Step 1.3**: Initialize Z, a vector with the best solutions for each subproblem.
- **Step 2: Update** For $i = 1, \cdots, P$ do
  - **Step 2.1**: Reproduction. Select 2 solutions and generate a new solution based on genetic operators.
  - **Step 2.2**: Improve the new solution if needed.
  - **Step 2.3**: Update neighbourhood solutions. (*Evaluating the models using repeated kcv with* S^1)
- **Step 3: Stopping criteria.**
- **Step 4: Select the Pareto front.** Initialize the set of non-dominated solutions PF.

  For all the generated solutions do
  - Add the solution to PF if no solution in PF dominates it. (*Evaluating the models using hold out with* S^1 *as the training set and* S^2 *as the test set*)
  - Remove from PF all the solutions dominated by the new solution.
- **Return** PF

Fig. 6.2. The modified MOEA/D algorithm. The modifications are in italics.

Then, in order to select the classifiers that make up the Pareto set at each selection step (step 4 on Figure 6.2), we estimate their accuracies using a Hold-out estimator with $\mathcal{S}^1$ as the training set and $\mathcal{S}^2$ as the test set. This way, the selection of the classifiers that form the Pareto front is independent of the structural learning process (See Figure 6.3). Recent papers have developed strategies to avoid overfitting when learning classifiers by means of multi-objective evolutionary learning approaches (Radtke et al., 2009; Dos Santos et al., 2009). These works defend the fact that the most appropriate approach is to validate candidate solutions in all generations during the learning process with a selection data set independent of that used in the optimization process. This strategy is called *global validation* and it is similar to that used in this work.

6.2.3 Codification of multi-dimensional Bayesian classifiers

To carry out the search procedure, each possible multi-dimensional Bayesian network structure is codified by a vector formed by three parts (see Figure 6.4):

- The first part is a permutation of the class variables $\mathbf{V_C}$ and represents an ancestral order over them. Its size is equal to the number of class variables.

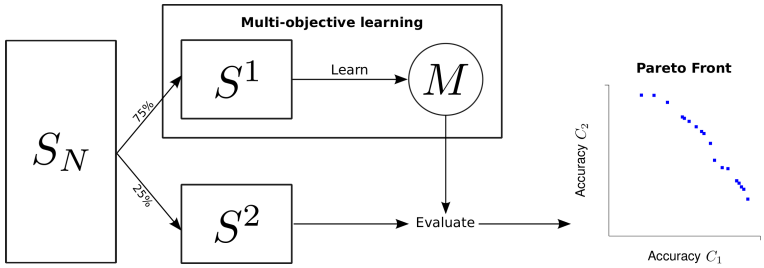


Fig. 6.3. Multi-objective Learning process.

- The second part is a binary vector that represents all the possible arcs of the bridge subgraph ($\mathbf{A_{CX}}$). Its size is equal to the number of class variables times the number of predictive variables.
- The third part is a permutation of the features $\mathbf{V_X}$ and represents an ancestral order over them. Its size is equal to the number of predictive variables.

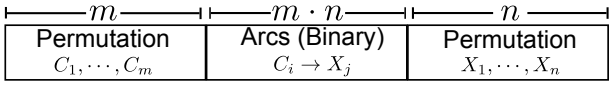


Fig. 6.4. Codification of an individual

The encoding does not explicity show all the arcs among the random variables (it only explicity encodes the arcs among class variables and features) because there is a combinatorial burden as the number of genes in the codification of an individual increases as the size of the individual is given by $m + m \cdot n + n$. The proposed encoding obtains the arcs among the class variables and among the predictive variables by means of statistical tests. Next we show how, given an individual, it is decoded into a multi-dimensional Bayesian network classifier structure using hypothesis tests.

We need to perform statistical tests in order to decide which arcs are included in the model. The chosen α of the used test has been set to 0.1 in order to avoid computational problems (the classifiers can end up with a high number of parameters). With a lower value of α we get classifiers with few arcs among the variables.

Given an individual, it is decoded into a multi-dimensional Bayesian network classifier structure as follows:

- The first permutation part represents an ancestral order in $\mathbf{V_C}$. Therefore, each class variable can be a parent of its successors in the ancestral order. In order to determine the set of arcs in the class subgraph ($\mathbf{A_C}$), we carry out a statistical test to determine the independence between the variables.

We only keep in the model the dependences with the p-values smaller than the threshold α in a independence test for each class variable.

- The binary part represents the arcs in the feature selection subgraph ($\mathbf{A_{CX}}$). A value of 1 represents an arc between a class variable and a feature.
- The last permutation part represents an ancestral order in $\mathbf{V_X}$. We let each predictive variable be a parent of its successors in the ancestral order. In order to determine the set of arcs in the class subgraph ($\mathbf{A_F}$), we carry out a statistical test to determine the independence between the variables. We only keep in the model the dependences with the p-values smaller than the threshold α in a independence test for each predictive variable.

The independence test we use to determine if a dependence between two variables is strong enough to be part of the model is based on the mutual information between them: It is known (Kullback, 1959) that $2N\hat{I}(X_i, X_j)$, if X_i and X_j are independent, asymptotically follows a χ^2 distribution with $(r_i - 1)(r_j - 1)$ degrees of freedom where N is the number of cases, X_i and X_j are random variables and r_i and r_j are the cardinality of X_i and X_j respectively: $lim_{N\to\infty} 2N\hat{I}(X_i, X_j) \rightsquigarrow \chi^2_{(r_i-1)(r_j-1)}$.

Based on this result, a null hypothesis test can be carried out in a multi-dimensional Bayesian network classifier to check the possible dependences in $\mathbf{A_C}$. The null hypothesis H_0 is: "the random variables C_i and C_j are independent". So, if the quantity $2N\hat{I}(C_i, C_j)$ surpasses a threshold s_α for a given test size $\alpha = \int_{s_\alpha}^{\infty} \chi^2_{(t_i-1)(t_j-1)} ds$, where t_i is the cardinality of C_i, and t_j the cardinality of C_j, the null hypothesis is rejected and there is a dependence between C_i and C_j. Therefore the arc between C_i and C_j is included in the model. This test was used on one-class Bayesian network classifiers to check the dependences among the class variables and the features (Blanco, 2005).

We use the conditional mutual information between a feature X_i and a feature X_j given its parents $\mathbf{Pa}(X_j)$ to determine if the arc between two features X_i and X_j in $\mathbf{A_F}$ should be included in the model. The previous result was generalized to the case of conditional mutual information as follows (Kullback, 1959):

$$lim_{N\to\infty} 2N\hat{I}(X_i, X_j|\mathbf{Pa}(X_j)) \rightsquigarrow \chi^2_{(r_i-1)(r_j-1)(|\mathbf{Pa}(X_j)|)}$$

where r_i is the cardinality of X_i, r_j the cardinality of X_j and $|\mathbf{Pa}(X_j)|$ the cardinality of the parents of X_j.

Analogously to the hypothesis test described before, based on these results we can perform the following conditional independence test. The null hypothesis H_0 is: "the random variables X_i and X_j are conditionally independent given $\mathbf{Pa}(X_j)$". So, if the quantity $2N\hat{I}(X_i, X_j|\mathbf{Pa}(X_j))$ surpasses a threshold s_α for a given test size $\alpha = \int_{s_\alpha}^{\infty} \chi^2_{(r_i-1)(r_j-1)(|\mathbf{Pa}(X_j)|)} ds$, the null hypothesis is

rejected and the random variables X_i and X_j are considered dependent given $\mathbf{Pa}(X_j)$. Therefore the arc is included in the model.

For example, given a problem with 3 class variables and 4 predictive variables, the following individual $(3, 1, 2|1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0|2, 1, 3, 4)$ is decoded to a multi-dimensional Bayesian network classifier in the following way:

We use the first permutation part, formed by $(3, 1, 2, \ldots)$, in order to build $\mathbf{A_C}$. It represents the ancestral order (C_3,C_1,C_2). So, there are 3 possible dependences $C_3 \rightarrow C_1$, $C_3 \rightarrow C_2$ and $C_1 \rightarrow C_2$. For each dependence, if the p-value of the statistic used in the independence test is smaller than the given significance level α, the arc is included in the model. We can suppose that in this case the arcs included in the model are the one from C_3 to C_2 and the one from C_1 to C_2.

The second part is a binary vector formed by $(\ldots, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, \ldots)$ and represents the arcs among the class variables and the features, that is, $\mathbf{A_{CX}}$. There are four features and three class variables, so the first 4 positions of this part $(\ldots, 1, 0, 1, 0, \ldots)$ represent the arcs between the first class variable C_1 and the features, the following 4 positions $(\ldots, 0, 1, 1, 1, \ldots)$ the arcs of the second class variable C_2 with the features and the last 4 positions $(\ldots, 0, 1, 1, 0, \ldots)$ the arcs between the third class variable C_3 and the features. A value of 1 in a position represents an arc. In this case there are arcs from C_1 to X_1 and X_3, from C_2 to X_2, X_3 and X_4 and from C_3 to X_2 and X_3. Finally, we have to build A_X. In order to do this, we use the last permutation part formed by $(\ldots, 2, 1, 3, 4)$. This represents the ancestral order (X_2, X_1, X_3, X_4). So, there are 6 possible dependences $X_2 \rightarrow X_1$, $X_2 \rightarrow X_3$, $X_2 \rightarrow X_4$, $X_1 \rightarrow X_3$, $X_1 \rightarrow X_4$ and $X_3 \rightarrow X_4$. For each dependence, if the p-value of the statistic used in the independence test is smaller than the given significance level α, the arc is included in the model.

X_2 has no parents, because it is the first feature in the ancestral order. X_1 can have X_2 as its parent, but in this case we suppose that its p-value is bigger than α. X_3 has X_2 and X_1 as parents because both p-values are smaller than α. X_4 can have X_2, X_1 and X_3 as parents. We suppose that the p-values of the arcs from X_2 and X_3 are smaller than α, so, these dependences are included in $\mathbf{A_X}$. The obtained classifier structure is given in Figure 6.5.

Finally, once we have fixed the structure, the parameters are learnt by maximum likelihood.

6.3 Experimentation

In this section, we present the proposed experimentation in order to evaluate our learning approach to multi-dimensional Bayesian network classifiers. The objective of the experimentation is to compare the proposed learning approach with other approaches to multi-dimensional classification.

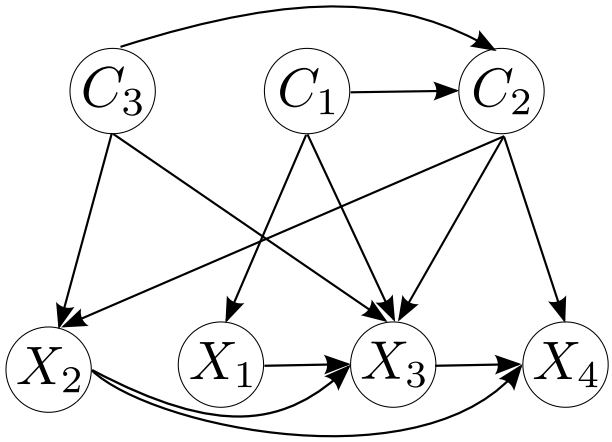


Fig. 6.5. The multi-dimensional Bayesian network classifier encoded in the individual $(3, 1, 2|1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0|2, 1, 3, 4)$.

6.3.1 Experimental set-up

We use the MOEA/D algorithm as the learning engine of our multi-objective approach. The parameters used for the experiments with this algorithm were set as follows:

- Number of subproblems: $P = 100*$(individual size).
- Number of neighbours for each subproblem: $T = 20$.
- Stop criterion: 100.000 evaluations.
- Number of replacements in the neighbourhood: $nr = 2$.
- Objective functions: We use the 5 repeated 5-fold stratified cross-validation error estimation of each class variable.

So as to make all these experiments possible, we use different open source libraries in Java. For the classification utilities we use the DMLIB library and the Weka library (Witten and Frank, 2000), for the multi-objective optimization utilities we use the jMetal library (Durillo et al., 2006) and for the multi-label utilities we use the Mulan library (Tsoumakas et al., 2011).

6.3.1.1 Evaluation of the Pareto front

The results of the multi-objective learning are presented in a Pareto front composed of different multi-dimensional classifiers that are non-dominated among them. Each point of the Pareto front represents the accuracies for each class of a specific classifier.

As we pointed out in Section 6.2.2, the MOEA/D algorithm has been modified in order to estimate the accuracies of the set of classifiers obtained by the proposed learning approach. Because of this modification, part of the

data is used to learn the model and the other part to test it. However, it should be clear that, when the classifiers are applied over unseen data, the models should be trained using all the available data. So, in order to compare the proposed approach with the selected benchmarks, we have made an external evaluation of the structures generated by MOEA/D.

For that issue, we have kept a part of the original data set unseen: $\mathcal{S}^3$, with 1/3 of the instances in $\mathcal{S}_N$. The rest of the data ($\mathcal{S}^1$ and $\mathcal{S}^2$) is used internally to learn the classifiers ($\mathcal{S}^1$ to learn the model and $\mathcal{S}^2$ to test it). Then, the external evaluation is made by means of a hold-out procedure using $\mathcal{S}^1 \bigcup \mathcal{S}^2$ to calculate the parameters of the classifiers and $\mathcal{S}^3$ to test them with unseen instances (see Figure 6.6).

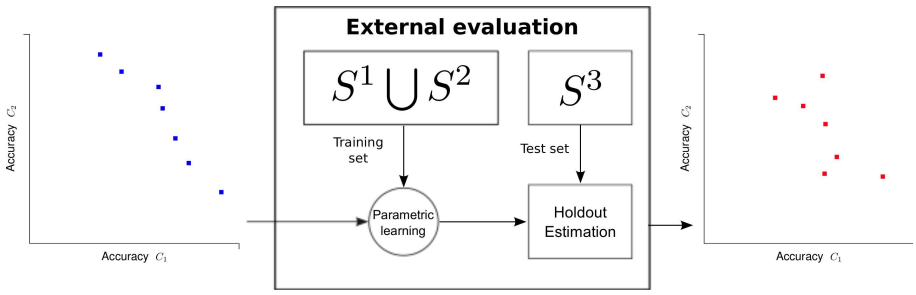


Fig. 6.6. External evaluation of the classifiers in the Pareto front.

Note that the classifiers of the Pareto front are re-evaluated externally and the final accuracy values could differ from the original ones. Thus, the re-evaluated solution set may not form a set of non-dominated solutions (see Figure 6.7), however, we can form a Pareto front with the solutions that still remain non-dominated among them.

In order to compare the proposed approach with the benchmark classifiers, we use the three different accuracy-based performance measures introduced in Section 3.3: The single, mean and global accuracies. As we use a multi-objective approach and it returns a Pareto front with several classifiers, we have selected the best classifier of the Pareto front on each measure to perform the comparisons (see Figure 6.8).

6.3.1.2 Benchmark classifiers

In order to compare the results of our approach, we have selected the approaches to multi-dimensional classification presented in Section 2.2.2:

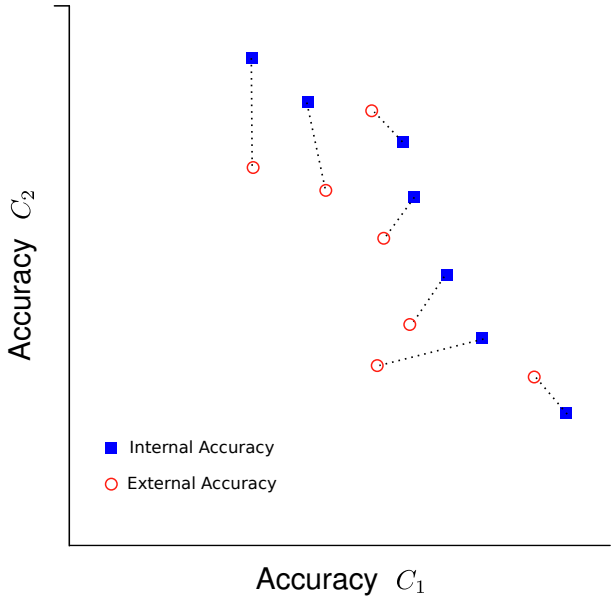


Fig. 6.7. The accuracy of the classifiers in the Pareto front tested internally and externally.

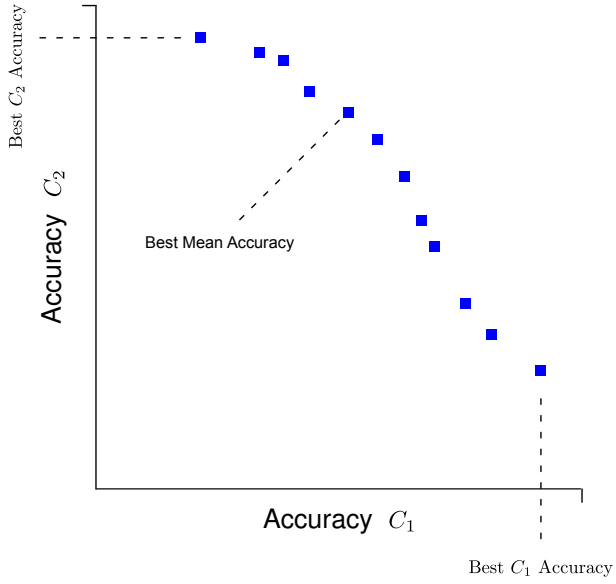


Fig. 6.8. Best accuracies of a Pareto front in a 2D domain.

- Single-objective learning approaches to MDBCs: We use the approaches proposed by van der Gaag and de Waal (2006) and de Waal and van der Gaag (2007) (MDnB and MDTAN, see Section 5.2) [1].

[1] Remember that MDnB and MDTAN obtain only one classifier and not a Pareto set

- Compound class variable: We have chosen a naïve Bayes classifier (cnB) (Langley et al., 1992; Minsky, 1961), a tree-augmented Bayesian classifier (cTAN) (Friedman et al., 1997) and a Support Vector Machine classifier (cSVM) (Cristianini, 2010; Burges, 1998).
- Multiple classifiers: We have used the same one-class classifiers used in the compound class variable approach (mnB, mTAN, mSVM). For each data set, we train m one-class classifiers (m nB, m TAN and m SVM), one for each class variable.
- Multi-Label classification: In Section 2.2.2 we explained how to transform a ML problem into a multi-dimensional problem, so a multi-dimensional classifier can be used to solve it. This way the results of using a multi-dimensional approach versus the use of a ML approach can be compared. We use a Multi-Label k Nearest Neighbours classifier (MLkNN) (Zhang and Zhou, 2007) and a Combined Instance-based Learning and Logistic Regression classifier (IBLR-ML) (Cheng and Hullermeier, 2009). In both algorithms we take into account 10 neighbours.

6.3.1.3 Benchmark data sets

In order to evaluate the proposed learning approach, we use some artificial and real-world multi-dimensional data sets.

The artificial data sets are sampled from multi-dimensional feature-label probability distributions $p(\boldsymbol{x}, \boldsymbol{c})$ represented as multi-dimensional Bayesian network classifiers. These Bayesian network classifiers have been created in two steps. First, the structure of the multi-dimensional Bayesian network classifier was created with the Java Bayes software (Cozman, 2000) and then the parameters are obtained sampling a Dirichlet distribution with all the parameters equal to one.

We have chosen the following structures:

- *MDTAN*
- *MDJ/K* with $J = 1$ and $K = 1$.
- *MDJ/K* with $J =$(number of classes)-1 and $K = 4$.

The number of predictive variables has been fixed to 15 and the number of class variables ranges from 2 to 6. The cardinality of the predictive variables ranges from 2 to 3. The cardinality of all the class variables has been set to 2, so the problem can be seen as a multi-label problem and compared straightforwardly with multi-label classification algorithms. Specifically, we have sampled 150 artificial data sets each of 500 instances (10 for the 3 different structures and 5 different number of class variables).

We have also used 2 different multi-dimensional real-world data sets. We have selected one multi-dimensional data set (automobile) whose class variables have more than 2 values and a multi-label data set (emotions).

- *Automobile data set* (Asuncion and Newman, 2007): This data set has 205 instances. Several variables can be considered as class variables so, we have

generated two different multi-dimensional classification problems, on one hand, one problem considers 2 variables as classes (symboling and price) and 24 as predictive variables, and on the other hand, another problem considers 4 variables as classes (city-mpg, highway-mpg, symboling and price) and 22 as predictive variables. Some variables are continuous and others discrete and there are missing values in some instances. The continuous variables have been discretized to 2 values using the equal frecuency method. We have eliminated the predictive variables that are missing in the majority of the instances (2 predictive variables) and in addition, we have eliminated the instances with missing values(13 instances).

- *Emotions data set* (Trohidis et al., 2008): This data set has 593 instances. There are 72 predictive variables and 6 binary class variables. All the predictive variables are continuous and have been discretized to 3 values using equal frecuency.

6.3.2 Results

In this section, we compare the multi-objective learning approach (MOP) proposed in this paper with the approaches to multi-dimensional classification mentioned in Section 6.3.1.2. The comparisons have been performed across all the generated data sets following the metodology introduced at the begining of Section 6.3.1.1.

We use statistical tests in order to know when there are statistical differences among the results presented in this section. Specifically, we have used a Friedmand test (Demsar, 2006) with a Shaffer's static post-hoc test (García and Herrera, 2008) both with $\alpha = 0.05$. The test results can be represented by means of critical difference diagrams, which show the mean ranks of each classification rule across all the domains in a numbered line. If there is no statistically significant difference between two methods, they are connected in the diagram by a straight line (see figures 6.9, 6.10 and 6.11) (Demsar, 2006).

6.3.2.1 Results on artificial data sets

In this section, we show the analysis of the results in the artificial data sets. First, we analyze the average of the mean and global accuracies of the studied classifiers across all the domains for all the possible numbers of class variables. Finally, we analyze the single accuracies separately for each class variable. In this case, we show the single accuracy for each possible number of class variables, as there are slight differences among them.

Table 6.1 shows the average mean and global accuracies of the studied classifiers across all the domains. It seems that the proposed multi-objective learning approach (MOP) reaches the best mean accuracy value with statistical differences with the rest of approaches (See Figure 6.9). On the other

extreme, the statistically significant worst classifiers in terms of mean accuracy are IBLR-ML and MLkNN, with statistical diferences between them. The rest of the classifiers (mTAN, cTAN, mnB, mSVM, MDTAN, cnB, cSVM and MDnB) are better than IBLR-ML and MLkNN and worse than MOP in terms of mean accuracy, but there are no statistical significative differences among them (See Figure 6.9).

Table 6.1. Mean and global accuracy. The best values are in bold.

		Mean acc		Global acc	
		Acc	**sd**	**Acc**	**sd**
MDBNC	**MOP**	**0,77160**	1,67E-2	0,31305	3,99E-2
	MDnB	0,70743	1,92E-2	0,34814	3,35E-2
	MDTAN	0,72117	2,27E-2	0,35257	3,73E-2
Compound	**nB**	0,71648	1,92E-2	0,37677	3,13E-2
	TAN	0,73630	2,01E-2	**0,41537**	3,55E-2
	SVM	0,71266	1,92E-2	0,37461	3,40E-2
Mutiple	**nB**	0,72216	1,98E-2	0,33158	3,61E-2
	TAN	0,74147	1,91E-2	0,36691	3,54E-2
	SVM	0,72213	1,86E-2	0,32423	3,42E-2
ML	**MLkNN**	0,69705	2,01E-2	0,30140	3,40E-2
	IBLR-ML	0,70796	2,07E-2	0,30407	3,54E-2

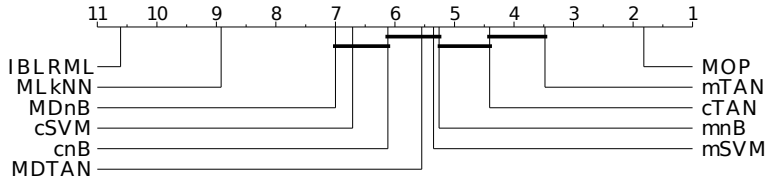


Fig. 6.9. Mean accuracy ranking.

If we focus on the global accuracy values of the studied classifiers, we notice that the use of a compound class variable approach with a TAN classifier (cTAN) leads to the best values with statistical significance (see Figure 6.10). The statistically worst global accuracy values are again those given by IBLR-ML and MLkNN classifiers. The rest of the approaches (cSVM, mTAN, cnB, MDnB, MDTAN, mnB, mSVM and MOP) are significantly better than IBLR-ML and MLkNN and worse than cTAN, but there are no statistical differences among them (see Figure 6.10).

We have also studied the best single accuracies for each class variable. Tables 6.2, 6.3, 6.4, 6.5 and 6.6 show the best accuracies for each class variable in the domains with 2, 3, 4, 5 and 6 classes respectively. In the domain with

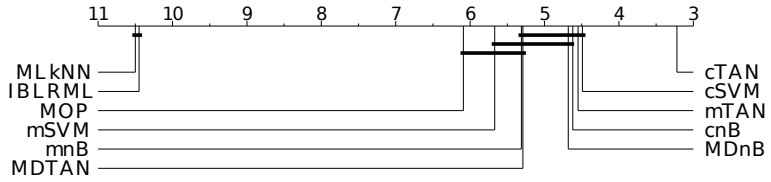


Fig. 6.10. Global accuracy ranking.

2 class variables, it seems that the use of a TAN classifier with a compound class variable approach (cTAN) and the use of the proposed approach (MOP) come up to the best single accuracy values. However, as long as we increase the number of class variables, the multi-objective learning approach gets better accuracy values and beats the rest of the approaches for all class variables.

Table 6.2. Single accuracy in 2-classes artificial domains. The best values are in bold.

		C1		C2	
		Acc	**sd**	**Acc**	**sd**
MOP	**Best** C_1	0,74850	1,42E-3	0,76747	1,38E-3
	Best C_2	0,71158	3,01E-3	**0,80379**	7,90E-4
MDBNC	**MDnB**	0,70958	3,44E-2	0,78423	2,98E-2
	MDTAN	0,74750	1,50E-3	0,76866	1,47E-3
Compound	**nB**	0,71178	3,84E-2	0,78383	3,15E-2
	TAN	**0,75948**	1,10E-3	0,79381	6,17E-4
	SVM	0,71138	1,40E-3	0,77784	6,48E-4
Multiple	**nB**	0,70060	4,04E-2	0,78762	3,53E-2
	TAN	0,75868	1,18E-3	0,79601	6,76E-4
	SVM	0,70419	8,47E-4	0,78862	7,54E-4
ML	**MLkNN**	0,70639	3,59E-2	0,73593	3,92E-2
	IBLR-ML	0,68862	3,51E-2	0,77325	3,55E-2

Figure 6.11 shows the average ranking for the single accuracy of the studied classifiers for all the domains and all the possible number of class variables. For the sake of brevity, we show the ranking for all the domains together. As expected, the results are very similar to the mean accuracy. The proposed approach is the most accurate and mTAN classifiers the second most accurate, with statistical differences among them and with the rest of the approaches. On the other side of the ranking, IBLR-ML and MLkNN return the worst single accuracy values with no statistical significances among them.

Table 6.3. Single accuracy in 3-classes artificial domains. The best values are in bold.

		C_1		C_2		C_3	
		Acc	**sd**	**Acc**	**sd**	**Acc**	**sd**
MOP	**Best** C_1	**0,8226**	5,72E-4	0,8028	1,75E-3	0,7465	1,58E-3
	Best C_2	0,7705	2,20E-3	**0,8417**	7,70E-4	0,7553	1,34E-3
	Best C_3	0,7655	9,96E-4	0,7878	1,73E-3	**0,7878**	6,87E-4
MDBNC	**MDnB**	0,7774	2,59E-2	0,7790	3,40E-2	0,7250	4,25E-2
	MDTAN	0,7766	1,03E-3	0,8032	9,62E-4	0,7539	1,37E-3
Compound	**nB**	0,7862	2,89E-2	0,7960	2,75E-2	0,7459	3,42E-2
	TAN	0,8126	9,95E-4	0,8152	1,06E-3	0,7782	9,06E-4
	SVM	0,7768	1,02E-3	0,7942	1,09E-3	0,7469	1,31E-3
Mutiple	**nB**	0,7780	2,71E-2	0,7912	3,23E-2	0,7547	3,39E-2
	TAN	0,8212	8,16E-4	0,8116	6,93E-4	0,7653	1,31E-3
	SVM	0,7677	7,14E-4	0,7852	1,61E-3	0,7573	9,75E-4
ML	**MLkNN**	0,7393	2,81E-2	0,7535	3,98E-2	0,7443	3,98E-2
	IBLR-ML	0,7617	3,04E-2	0,7804	3,92E-2	0,7517	4,08E-2

Table 6.4. Single accuracy in 4-classes artificial domains. The best values are in bold.

		C_1		C_2		C_3		C_4	
		Acc	**sd**	**Acc**	**sd**	**Acc**	**sd**	**Acc**	**sd**
MOP	**Best** C_1	**0,7493**	9,72E-4	0,7186	2,14E-3	0,7192	8,39E-4	0,6749	1,72E-3
	Best C_2	0,6820	1,66E-3	**0,7752**	6,63E-4	0,7206	1,20E-3	0,7048	1,26E-3
	Best C_3	0,6758	2,13E-3	0,7230	1,85E-3	**0,7790**	7,49E-4	0,6952	1,64E-3
	Best C_4	0,6663	1,50E-3	0,7299	1,21E-3	0,7277	1,25E-3	**0,7469**	5,90E-4
MDBNC	**MDnB**	0,6693	3,35E-2	0,7042	2,64E-2	0,7080	3,63E-2	0,6836	4,03E-2
	MDTAN	0,6900	1,76E-3	0,6926	1,20E-3	0,7228	2,36E-3	0,6856	1,11E-3
Compound	**nB**	0,6930	3,72E-2	0,7214	2,81E-2	0,7255	3,21E-2	0,6952	3,68E-2
	TAN	0,7248	1,64E-3	0,7339	1,21E-3	0,7415	1,07E-3	0,7150	1,08E-3
	SVM	0,6882	1,62E-3	0,7164	1,03E-3	0,7240	8,16E-4	0,6956	1,51E-3
Mutiple	**nB**	0,6884	3,80E-2	0,7351	3,09E-2	0,7403	3,28E-2	0,6956	3,62E-2
	TAN	0,7236	1,61E-3	0,7353	1,01E-3	0,7397	1,45E-3	0,7018	1,40E-3
	SVM	0,6858	1,78E-3	0,7327	1,05E-3	0,7365	1,26E-3	0,7032	1,95E-3
ML	**MLkNN**	0,6599	3,51E-2	0,7164	3,30E-2	0,7307	3,90E-2	0,6752	3,91E-2
	IBLR-ML	0,6661	3,81E-2	0,7281	3,70E-2	0,7367	3,59E-2	0,6739	3,60E-2

Table 6.5. Single accuracy in 5-classes artificial domains. The best values are in bold.

		C_1		C_2		C_3		C_4		C_5	
		Acc	**sd**	**Acc**	**sd**	**Acc**	**sd**	**Acc**	**sd**	**Acc**	**sd**
MOP	**Best** C_1	**0,7293**	7,35E-4	0,6563	1,84E-3	0,7647	1,19E-3	0,6547	2,43E-3	0,7024	1,87E-3
	Best C_2	0,6439	1,23E-3	**0,7369**	5,16E-4	0,7661	7,11E-4	0,6539	2,74E-3	0,7004	1,73E-3
	Best C_3	0,6339	1,53E-3	0,6423	1,88E-3	**0,8158**	6,00E-4	0,6565	1,45E-3	0,6852	1,89E-3
	Best C_4	0,6425	2,04E-3	0,6455	1,81E-3	0,7689	5,98E-4	**0,7527**	1,20E-3	0,6940	2,32E-3
	Best C_5	0,6535	1,92E-3	0,6583	1,15E-3	0,7661	1,04E-3	0,6523	2,17E-3	**0,7681**	6,49E-4
MDBNC	**MDnB**	0,6361	4,03E-2	0,6581	3,34E-2	0,7533	3,22E-2	0,6589	3,42E-2	0,6886	4,00E-2
	MDTAN	0,6451	2,88E-3	0,6417	2,66E-3	0,7719	8,44E-4	0,7004	2,00E-3	0,7116	1,68E-3
Compound	**nB**	0,6457	4,13E-2	0,6549	3,39E-2	0,7587	3,10E-2	0,6693	3,52E-2	0,6986	4,22E-2
	TAN	0,6543	1,45E-3	0,6780	1,02E-3	0,7623	7,27E-4	0,7056	1,10E-3	0,7066	1,48E-3
	SVM	0,6383	1,57E-3	0,6561	1,74E-3	0,7599	7,28E-4	0,6643	8,65E-4	0,7020	1,66E-3
Mutiple	**nB**	0,6709	3,90E-2	0,6796	3,38E-2	0,7709	2,74E-2	0,6750	3,96E-2	0,7160	3,58E-2
	TAN	0,6754	8,11E-4	0,6840	1,51E-3	0,7691	1,36E-3	0,7297	1,09E-3	0,7265	1,19E-3
	SVM	0,6729	1,70E-3	0,6683	1,44E-3	0,7802	8,40E-4	0,6788	1,51E-3	0,7174	1,32E-3
ML	**MLkNN**	0,6263	2,99E-2	0,6441	3,84E-2	0,7802	2,78E-2	0,6451	3,90E-2	0,6569	2,97E-2
	IBLR-ML	0,6421	3,48E-2	0,6605	3,47E-2	0,7701	2,78E-2	0,6391	4,10E-2	0,6934	4,60E-2

Table 6.6. Single accuracy in 6-classes artificial domains. The best values are in bold.

		C_1		C_2		C_3		C_4		C_5		C_6	
		Acc	**sd**	**Acc**	**sd**	**Acc**	**sd**	**Acc**	**sd**	**Acc**	**sd**	**Acc**	**sd**
MOP	**Best** C_1	**0,7609**	6,31E-4	0,6567	1,88E-3	0,6818	2,85E-3	0,6024	9,54E-4	0,6271	1,31E-3	0,6846	1,60E-3
	Best C_2	0,6802	1,41E-3	**0,7475**	6,50E-4	0,6982	3,33E-3	0,5934	9,17E-4	0,6347	1,50E-3	0,6842	1,83E-3
	Best C_3	0,6910	1,67E-3	0,6637	2,34E-3	**0,7826**	1,09E-3	0,5962	1,11E-3	0,6357	1,72E-3	0,6780	1,27E-3
	Best C_4	0,6882	1,42E-3	0,6607	1,58E-3	0,6808	2,32E-3	**0,6749**	4,50E-4	0,6355	1,10E-3	0,6854	1,14E-3
	Best C_5	0,6970	1,24E-3	0,6792	1,03E-3	0,6838	1,61E-3	0,6072	1,22E-3	**0,7220**	6,46E-4	0,6788	2,01E-3
	Best C_6	0,6988	1,23E-3	0,6543	3,03E-3	0,6812	2,61E-3	0,6076	1,30E-3	0,6303	1,93E-3	**0,7599**	5,07E-4
MDBNC	**MDnB**	0,6996	3,51E-2	0,6645	3,73E-2	0,6824	3,76E-2	0,5928	4,04E-2	0,6403	4,56E-2	0,6774	3,20E-2
	MDTAN	0,6966	9,59E-4	0,6822	1,10E-3	0,7146	3,81E-3	0,6164	1,42E-3	0,6567	1,48E-3	0,7012	1,30E-3
Compound	**nB**	0,6998	3,73E-2	0,6750	2,76E-2	0,6872	3,82E-2	0,6000	3,93E-2	0,6471	3,74E-2	0,6768	3,61E-2
	TAN	0,7002	1,16E-3	0,6772	7,07E-4	0,7124	1,23E-3	0,6098	1,29E-3	0,6619	1,56E-3	0,6747	1,06E-3
	SVM	0,6904	1,62E-3	0,6645	1,59E-3	0,6768	1,48E-3	0,5900	1,01E-3	0,6447	1,55E-3	0,6689	1,57E-3
Mutiple	**nB**	0,7168	3,77E-2	0,6782	3,78E-2	0,6874	3,38E-2	0,6058	4,07E-2	0,6447	3,69E-2	0,7152	2,76E-2
	TAN	0,7092	1,13E-3	0,6994	1,22E-3	0,7413	1,22E-3	0,6164	1,48E-3	0,6599	1,12E-3	0,7054	1,02E-3
	SVM	0,7122	1,40E-3	0,6796	1,99E-3	0,7000	1,56E-3	0,6078	1,47E-3	0,6425	1,87E-3	0,7144	8,28E-4
ML	**MLkNN**	0,6892	3,00E-2	0,6335	3,38E-2	0,6912	3,35E-2	0,5918	3,94E-2	0,6403	3,05E-2	0,6677	3,26E-2
	IBLR-ML	0,6976	3,71E-2	0,6641	3,50E-2	0,6701	3,68E-2	0,5886	4,03E-2	0,6411	3,63E-2	0,7108	3,28E-2

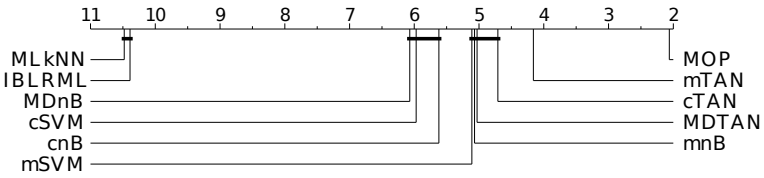


Fig. 6.11. Single accuracy ranking.

6.3.2.2 Results on real-world data sets

Finally, we have tested our learning approach on two real world data sets: Automobile and Emotions. Firstly, we analyze the results over the Automobile data set with 2 and 4 class variables. Its class variables are not binary, so it can not straightforwardly be compared with multi-label approaches.

Figure 6.12 shows the Pareto front obtained in the case of two classes with the proposed approach compared with MDnB, MDTAN and nB, TAN and SVM classifiers for the compound class variable approach (cnB, cTAN, cSVM) and multiple classifiers approach (mnB, mTAN, mSVM). The best accuracy values of each Pareto front for the 2 and 4 classes cases are compared with the rest of the approaches in Tables 6.7 and 6.8. The best mean values are shown in bold.

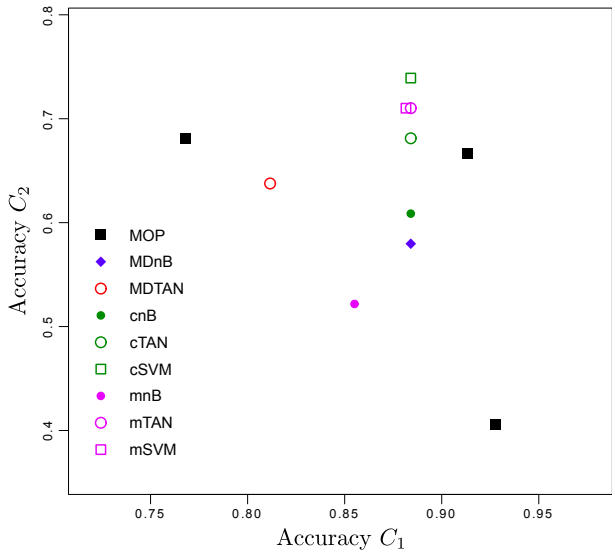


Fig. 6.12. Accuracies of the different approaches to the Automobile data set with 2 class variables.

In the case of two classes, the proposed approach comes up to the best single accuracy for class C_1, but the use of a SVM classifier with a compound class variable approach (cSVM) seems to have the best accuracy value for C_2. cSVM also has the best mean accuracy and the MOP approach achieves the second best mean accuracy. In the case of four classes, the MOP approach reaches the best single accuracy for all class variables and it also leads to the best mean accuracy. cSVM and mTAN also reach the best single accuracy for C_1 and C_3 respectively. If we focus on the global accuracy, that is, guess all the class variables simultaneously, we show that the MOP approach is not very competitive. That seems to be an expected result as our learning approach tries to optimize the single accuracy of each class variable.

Table 6.7. Accuracies for the Automobile data set with two class variables.

		C_1	C_2	**Mean acc**	**Global acc**
MDBNC	**MOP**	**0,9275**	0,6812	0,8043	0,6232
	MDnB	0,8841	0,5797	0,7319	0,4927
	MDTAN	0,8116	0,6377	0,7246	**0,7246**
Compound	**nB**	0,8841	0,6087	0,7464	0,5362
	TAN	0,8841	0,6812	0,7826	0,6812
	SVM	0,8841	**0,7391**	**0,8116**	0,6377
Multiple	**nB**	0,8551	0,5217	0,6884	0,4493
	TAN	0,8841	0,7101	0,7971	0,6087
	SVM	0,8816	0,7101	0,7966	0,5827

Table 6.8. Accuracies for the Automobile data set with four class variables.

		C_1	C_2	C_3	C_4	**Mean acc**	**Global acc**
MDBNC	**MOP**	**0.9538**	**0.9538**	**0.9692**	**0.7692**	**0.9115**	0.3385
	MDnB	0.9231	0.9231	0.9077	0.6307	0.8461	0.5076
	MDTAN	0.8923	0.9077	0.9077	0.5231	0.8077	0.4615
Compound	**nB**	0.9231	0.9231	0.9385	0.6153	0.8500	0.5385
	TAN	0.8769	0.8923	0.9231	0.6308	0.8308	0.5231
	SVM	**0.9538**	0.9385	0.9385	0.7384	0.9000	**0.6307**
Multiple	**nB**	0.8308	0.8615	0.9384	0.5231	0.7885	0.4308
	TAN	0,8769	0.8769	**0.9692**	0.6308	0.8385	0.5077
	SVM	0.9231	0.8923	0.9538	0.6923	0.8654	0.6154

Table 6.9 shows the results of the experiments in the Emotions data set. The classifiers obtained with the proposed approach only perform the best for the class variable C_4. The TAN based classifiers get the best results for the most of the class variables (MDTAN wins for C_2 and C_5 and mTAN wins for C_3 and C_6). The multi-label algorithm MLkNN gets the best accuracy value

for C_1 and C_5. However, the proposed MOP approach and mTAN have the best mean accuracy for the six class variables. The MOP approach clearly wins in global accuracy terms. It seems that, in this particular problem, the approach gets a better perfomance when guessing several classes simultaneously. The result seems expected, as it is a multi-dimensional problem with a high number of class variables. Moreover, the proposed approach returns a set of classifiers and we have chosen the best one.

Table 6.9. Accuracies for the Emotions data set.

		C_1	C_2	C_3	C_4	C_5	C_6	**Mean acc**	**Global acc**
MDBNC	**MOP**	0,7828	0,7071	0,7525	**0,8889**	0,7929	0,7525	**0,7795**	**0,3535**
	MDnB	0,7071	0,7424	0,6919	0,8586	0,6414	0,7020	0,7239	0,1364
	MDTAN	0,7273	**0,7778**	0,6919	0,8485	**0,8030**	0,7727	0,7702	0,2677
Compound	**nB**	0,7121	0,7475	0,7172	0,8687	0,6515	0,7121	0,7348	0,1869
	TAN	0,7727	0,7424	0,7121	0,8485	0,7424	0,7727	0,7651	0,2374
	SVM	0,7273	0,6515	0,7475	0,8586	0,7576	0,6919	0,7391	0,2172
Multiple	**nB**	0,7424	0,6263	0,7424	0,8586	0,7626	0,8232	0,7593	0,2323
	TAN	0,7525	0,6768	**0,7677**	0,8636	0,7828	**0,8333**	**0,7795**	0,2374
	SVM	0,7424	0,6919	0,6970	0,8434	0,7980	0,7323	0,7508	0,2252
ML	**MLkNN**	**0,8030**	0,7525	0,6919	0,8636	**0,8030**	0,7273	0,7736	0,1970
	IBLR-ML	0,7727	0,7475	0,7222	0,8690	0,7980	0,7071	0,7694	0,2020

6.4 Conclusions

In this chapter, we face the problem of learning Bayesian network classifiers for multi-dimensional supervised classification problems. To that end, we present a multi-objective learning approach to multi-dimensional Bayesian classifiers. This approach returns a Pareto set of non-dominated multi-dimensional Bayesian classifiers learnt from the same data set. The presented approach considers the learning of multi-dimensional Bayesian classifiers whose objective is to maximize the accuracy of each class variable, and it uses a multi-objective optimization algorithm (MOEA/D) to learn a set of non-dominated classifiers. We have used the 5 repeated 5-fold stratified cross-validation error estimation (5cv) of each class variable as objective functions for the multi-objective optimization problem, given its low variance (Rodríguez et al., 2010).

In this study, we compare the proposed approach using the joint classification rule with the main alternatives to deal with multi-dimensional classification (10 models in total). This analysis is made for artificial and real-world domains. In the artificial domains (see Section 6.3.2.1), the classifiers obtained with the proposed learning approach come up to the best mean accuracy and single accuracy for all the class variables, and it seems to improve its performance as the number of class variables increases. If we focus on the real-world data sets (see Section 6.3.2.2), we find no clear winner for the single accuracy in the two-classes domain, but as long as we increase the number of class variables to 4 we find that the proposed approach leads to the best single accuracy values. In the six-classes domain we found that several approaches get to the best single accuracy results (we have to take into account that this data set can be considered a multi-label data set). However, the proposed approach reaches the best mean accuracy in both the 4 and 6 classes domains. Finally, the proposed approach does not have the best global accuracy for the 2 and 4 classes domains but it reaches the best global accuracy in the 6 classes domain. To sum up, it seems that the proposed approach has a competitive behaviour, especially when the number of class variables involved is high.

Part IV

Application of Multi-dimensional Bayesian Network Classifiers on Medical Domains

7

Using Multi-Dimensional Bayesian Network Classifiers to Assist the Treatment of Multiple Sclerosis

Multiple sclerosis is an autoimmune disorder of the central nervous system and potentially the most common cause of neurological disability in young adults. The clinical disease course is highly variable and different multiple sclerosis subtypes can be defined depending on the progression of the severity of the disease. In the early stages, the disease subtype is unknown, and there is no information about how the severity is going to evolve. As there are different treatment options avaliable depending on the progression of the disease, early identification has become highly relevant. Thus, given a new patient, it is important to diagnose the disease subtype. Another relevant piece of information to predict is the expected time to reach a severity level indicating that assistance for walking is required.

Given that we have to predict two correlated class variables: Disease subtype and time to reach certain severity level, we can consider the use of multi-dimensional Bayesian network classifiers because they can model and exploit the relations among both variables. Besides, the obtained models can be validated by the physicians by using their expert knowledge due to the interpretability of Bayesian networks.

In the previous chapter we presented a multi-objective learning approach of multi-dimensional Bayesian network classifiers with no structural restrictions. In this chapter, we present a slight modification of that multi-objective learning approach that deals with the learning of MD$J/K$ Bayesian network classifiers. The application of the methodology proposed in this chapter can help a physician to identify the expected progression of the disease and to plan the most suitable treatment.

7.1 Introduction

Multiple sclerosis (MS) is potentially the most common cause of neurological disability in young adults. It is an autoimmune disease affecting the central

nervous system and it is characterized by demyelination and neurodegeneration. The progression of the disease and other phenotypic variability of MS seem to be influenced by genetic and enviromental factors (Frohman et al., 2005). The responsible genes are not mutations coding for aberrant genes, but normal polymorphisms. They act independently or through epistasis, and each polymorphism can exert a small contributory effect on some as yet undefined structure of physiological function. The course of the disease and the speed of the disability is variable. In about 25% of patients, MS never affects daily activities, but up to 15% of them become severely disabled and wheelchair-bound within a short period of time (Compston and Coles, 2002). Moreover, the clinical disease course depends on the type of MS: Some patients have a steady increase in disability while others suffer from episodes which may or may not leave permanent deficits, followed by periods of remission.

The disability status is usually determined using *Kurtzke's Expanded Disability Status Scale* (EDSS) (Kurtzke, 1983). This scale ranges from 0 to 10. A patient with no disability is 0, and the disability increases until level 6 but still with full ambulation. Between level 6 and 7 the patient needs aid to walk, and from level 7 on is unable to walk and is restricted to a wheelchair or bed. With level 9.5 the patient is unable to communicate effectively or eat/swallow and level 10 means death due to MS. There is a clinically relevant point, $EDSS = 6$, indicating that assistance for walking is required.

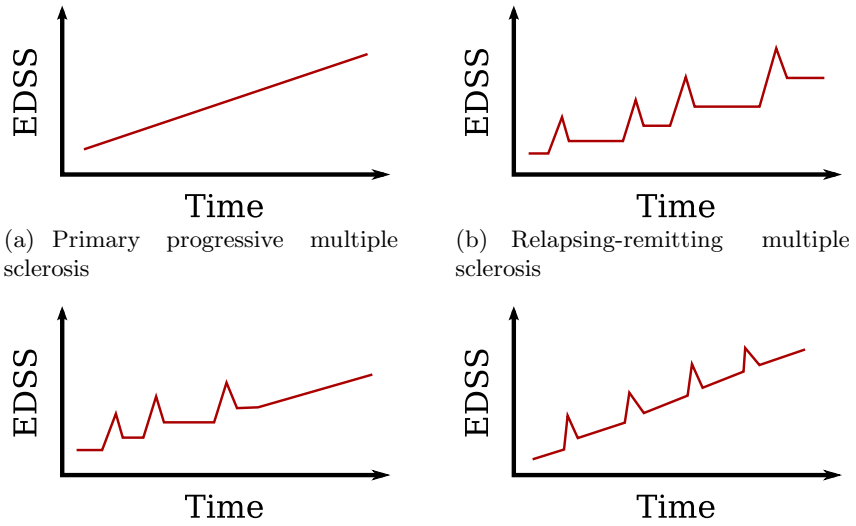
Given the clinical disability course through time, multiple sclerosis can be divided in four different types (see Figure 7.1). Note that they represent different qualitative types of the disease, not different severity grades:

- Primary progressive MS: The disease severity increases steadily without episodes.
- Relapsing-remitting MS: The disease is characterized by unpredictable attacks which may or may not leave permanent deficits, followed by periods of complete or partial remission.
- Secondary progressive MS: It follows the behaviour of relapsing-remitting MS in the initial steps of the disease and suddenly it starts to progress without periods of remission.
- Progressive-relapsing MS: The disease severity steadily increases from the onset with superimposed acute episodes and no relief from accumulated symptoms.

Different treatment options have become available with different efficacy and side effects, depending on the type and progression of the disease. Thus, the knowledge gained across a supervised classification approach could help to make the strategy for disease management more suitable.

There are two variables to be predicted that can help in this issue:

- The subtype of the disease: This shows the way the disability is going to evolve.



(a) Primary progressive multiple sclerosis

(b) Relapsing-remitting multiple sclerosis

(c) Secondary progressive multiple sclerosis

(d) Progressive-relapsing multiple sclerosis

Fig. 7.1. Different types of multiple sclerosis

- The time to reach $EDSS = 6$: This shows how fast the disability increases.

These variables seem to be correlated, as both define the progression of the disease, and could be considered as classes in a supervised classification approach. Given that there is more than one class variable and it seems that there is a correlation between these variables, we propose to approach the problem as an instance of the new paradigm of multi-dimensional classification (van der Gaag and de Waal, 2006; de Waal and van der Gaag, 2007; Rodríguez and Lozano, 2010; Bielza et al., 2011). In this chapter we construct MDBCs because they take into account the conditional dependences between the class variables. The use of these dependences should help in the classification task.

These classifiers use genetic information of a DNA chip and some clinical characteristics as predictive variables, and the *subtype of the disease* and the *time to reach* $EDSS = 6$ as class variables.

The MDBC used in this chapter is the *multi-dimensional* J/K *dependences Bayesian classifier* (Rodríguez and Lozano, 2010) with $J = 1$ and $K = 2$. This classifier is learnt by means of a multi-objective approach guided by the Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) (see Section 6.2.1).

As in the approach presented in the previous chapter, the use of a multi-objective learning approach returns a Pareto set of classifiers with different accuracy values for each of the classes. There are mainly two reasons to let

the physician select a classifier from a set of classifiers instead of returning only one classifier:

- Expert validation and selection: The interpretability of the multi-dimensional Bayesian network classifiers makes it possible to study the relationships among all the variables involved in the disease and lets the physician check whether they are correct or no (Lacave et al., 2007). The physician can then validate the classifiers using his/her expert knowledge, and select the most suitable in terms of the modeled relations.
- Selection based on the accuracy: The physician can select the most suitable classifier, giving relative importance to the accuracy of each class variable, using the mean accuracy or the global accuracy as the selection key. The doctor can even select the best classifier for each class variable in order to classify the patients.

The rest of the chapter is organized as follows: In Section 7.2 we introduce the multi-objective learning algorithm for multi-dimensional Bayesian network classifiers used in this chapter. Section 7.3 presents the experimental methodology and the obtained empirical results. Finally, the conclusions are presented in Section 7.4.

7.2 Learning J/K dependences Bayesian classifiers

The learning approach presented in this chapter is quite similar to the one presented in Chapter 6. The main difference is that in this application we use the learning approach to learn MD$J/K$ Bayesian network classifiers instead of unrestricted MDBCs. We use a restricted version of the proposed learning approach in order to control the computational cost of the model in a real-world data set.

In this chapter we use the same codification of the individuals used in the previous chapter. However, as in this chapter we learn a different kind of MDBC, the decodification of the individuals is changed. Given an individual, it is decoded into a multi-dimensional Bayesian network classifier structure as follows:

- The first permutation part represents an ancestral order in the set of class variables. Therefore, each class variable can be a parent of its successors in the ancestral order. In order to determine the set of arcs in the class subgraph, we carry out a independence test like that applied in Section 6.2.3 but only keep in the model a maximum of J dependences. Those dependences are, at the most, the J with the lowest p-values smaller than the threshold α for each class variable.
- The binary part represents the arcs in the feature selection subgraph. A value of 1 represents an arc between a class variable and a feature.

- The last permutation part represents an ancestral order in the set of features. We let each predictive variable be a parent of its successors in the ancestral order. The set of arcs of the feature subgraph is composed of a maximum of K dependences. Those dependences are, at the most, the K with the lowest p-values smaller than the threshold α for each predictive variable.

For example, given a problem with 2 class variables and 4 predictive variables, the following individual $(1, 2|1, 0, 1, 0, 0, 1, 1, 1|2, 1, 3, 4)$ is decoded to a MD1/2 Bayesian network classifier in the following way:

We use the first permutation part, formed by $(1, 2, \ldots)$, in order to build the arcs among the class variables. It represents the ancestral order (C_1,C_2). So, there is a possible arc between C_1 and C_2. If the p-value of the statistic used in the independence test is smaller than the given significance level α, the arc is included in the model. We can suppose that in this case the arc is included in the model. The second part is a binary vector formed by $(\ldots, 1, 0, 1, 0, 0, 1, 1, 1, \ldots)$ and represents the dependences among the class variables and the features. There are four features and two class variables, so the first 4 positions of this part $(\ldots, 1, 0, 1, 0, \ldots)$ represent the dependences between the first class variable C_1 and the features, and the following 4 positions $(\ldots, 0, 1, 1, 1, \ldots)$ the dependences of the second class variable C_2 with the features. A value of 1 in a position represents an arc. In this case there are dependences from C_1 to X_1 and X_3, and from C_2 to X_2, X_3 and X_4. Finally, we have to build the arcs among the features. In order to do this, we use the last permutation part formed by $(\ldots, 2, 1, 3, 4)$. This represents the ancestral order (X_2, X_1, X_3, X_4). For each variable, the K dependences with the lowest p-values of the statistic used in the independence test smaller than the threshold α are included in the model. X_2 has no parents, because it is the first feature in the ancestral order. X_1 can have X_2 as its parent, but in this case we suppose that its p-value is higher than α. X_3 has X_2 and X_1 as parents because their p-values in the statistic are smaller than α. X_4 can have X_2, X_1 and X_3 as parents. We suppose that the p-values of all the possible dependences are smaller than α, but we have a limit of K parents for each feature. So, only the 2 dependences with the lowest p-values are included in the model. In this case we suppose that these arcs are those from X_2 and X_3. The obtained classifier structure is given in Figure 7. Finally, once we have learnt the structure, the parameters are learnt by maximum likelihood.

7.3 Experimentation

7.3.1 The data set

The data set used is composed of DNA and clinical information of a total of 605 unrelated Dutch caucasian patients selected from natural history studies conducted at the MS Center at the VU University Medical Center (VUmc) in

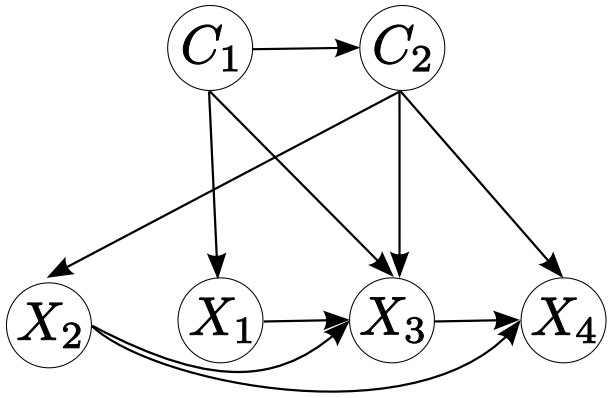


Fig. 7.2. The MD1/2 Bayesian network classifier encoded in the individual $(1, 2|1, 0, 1, 0, 0, 1, 1, 1|2, 1, 3, 4)$.

Amsterdam (The Netherlands[1]). The selection was based on the avaliability of DNA, clinical assessment of disability and the confirmed diagnosis of MS. No inclusion criteria for disability status, age, gender or onset type were applied during the selection of data for analysis. This study was carried out with the approval of the Medical Ethics Commitee of the VUmc and informed consent was obtained from all participant patients. This data set has been previously used in Sombekke et al. (2010) in order to detect correlations between the genetic/clinical data and the progression of the disease.

Each instance of the data set is composed of 86 features and 2 class variables. The features are 80 single nucleotide polymorphisms (SNPs) and 6 clinical variables (date of birth, gender, onset type, age at onset, onset year and $EDSS > 6$, a binary variable that represents whether the patient has reached $EDSS = 6$ or not). Physicians are particulary interested in the prediction of variables that characterize the progression of the disease such as the *subtype of MS* and the *time to reach* $EDSS = 6$.

Nine of the features are continuous and the rest are discrete. The cardinality of the discrete features ranges from 2 to 3. One class variable is discrete: The *subtype of MS* (primary progressive MS, relapsing-remitting MS, secondary progressive MS and progressive-relapsing MS), and the other variable, *time to reach* $EDSS = 6$, is continuous (in the following section we will see how to discretize it).

An interesting methodological point of this work consists of dealing with a temporal variable (lifetime), which, depending on the patient, provides partial or total information about the time elapsed to reach level 6 of EDSS. The next

[1] Any information related with this institution can be found at: http://www.vumc.com/

section focuses on introducing the treatment of *time to reach* $EDSS = 6$ class variable in our proposed approach.

7.3.2 Time to reach $EDSS = 6$

In this particular data set, there is a peculiarity with the class variable *time to reach* $EDSS = 6$. Some of the instances (about 2/3) are individuals that have not yet reached level 6 of EDSS. In these individuals, the variable *time to reach* $EDSS = 6$ does not represent what we are looking for, that is, the time that has passed since the individual was diagnosed until reaching level 6 of severity. In these instances, the class variable measures the time that has passed from being diagnosed to performing the last EDSS test. The only certainty given by these instances is that the individual has not reached $EDSS = 6$ at that moment. That is, the real *time to reach* $EDSS = 6$ value can be greater than or equal to but not lower than the current value. This variable can be considered as a special lifetime variable, since, depending on the patient, it measures different characteristics. Therefore, when the patients have not reached the $EDSS = 6$ level, different learning and testing procedures must be developed for this class variable.

In first place, we have discretized the class variable *time to reach* $EDSS = 6$ in order to face this problem using a supervised classification approach. This variable has been discretized into four intervals. The values are: Less than 5 years, between 5 and 10 years, between 10 and 20 years and more than 20 years.

In order to learn the classifiers, the instances associated to the patients that have not reached $EDSS = 6$ have been replicated taking into account all the *allowed* values that the variable *time to reach* $EDSS = 6$ can take. The *allowed* values are equal to or greater than the current value measured for the particular patient. Then, in order to maintain the relative importance of each patient in the training set, the replicated instances are equally weighted so that they sum one. This approach is an attempt to extract as much information as possible from the available data, preventing at the same time the introduction of biases into the learning step of our approach (see Figure 7.3).

We have also proposed a novel evaluation function in order to deal with the particular nature of the class variable *time to reach* $EDSS = 6$. The evaluation function is equivalent to the accuracy for the patients that have reached $EDSS = 6$. When we are classifying a patient that has not reached $EDSS = 6$, the function considers that the classification is correct if the predicted value is equal to or higher than the quantified value, and incorrect otherwise. This approach is appropriate for this problem taking into account the available data and the replication scheme used in the learning step.

In the following section, we explain the discretization of the continuous variables that have been performed and the attribute selection process carried out in order to reduce the number of implied variables. We also set the parameters of the multi-objective learning approach.

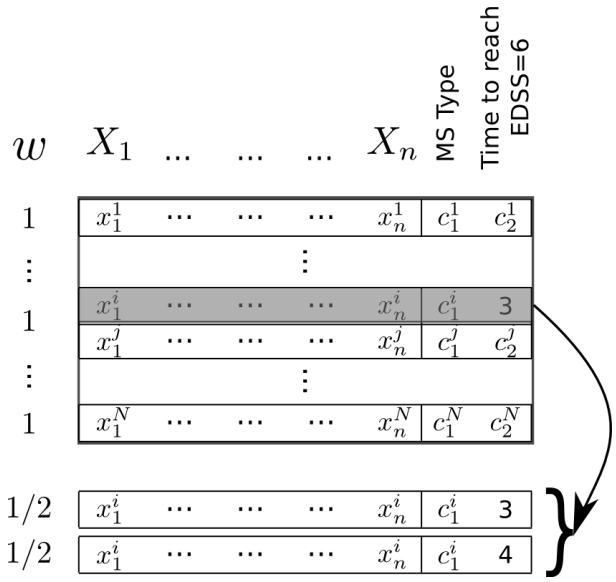


Fig. 7.3. Learning process of *time to reach* $EDSS = 6$. The shaded instance has not yet reached level 6 of EDSS.

7.3.3 Experimental setup

The initial step of the experimental setup is to preprocess the data set in order to properly apply multi-dimensional Bayesian classifiers. First of all, we have discretized the continuous variables. The nine continuous features are discretized to five values using equal-frecuency interval binning (Dougherty et al., 1995).

Once all the variables were discretized, we carried out an attribute selection process to reduce the number of features before learning a classifier because redundant variables can have a negative influence on the accuracy of classifiers based on Bayesian networks (Hall, 1999). We have used a multi-dimensional feature subset selection process based on the popular Correlation-based Feature subset Selection (CFS) score (Hall, 2000). The method we have used is the CFSindiv (Fernandes et al., 2013). It consists of selecting the union of all variable subsets that have been selected by the uni-dimensional CFS for each class variable in isolation. This method has selected 21 features, 17 of them SNPs and 4 clinical variables. The clinical variables are: Date of birth, onset type, age at onset and onset year.

The next step is to define the characteristics of the multi-dimensional Bayesian network classifiers. We use a MD$J/K$ Bayesian network classifier with $J = 1$ and $K = 2$, that is, a MD1/2. The multi-dimensional nature of the problem allows different classification rules that would make no sense in one-class classification (see Section 5.3). In this work, we will use the *joint*

classification rule, $\hat{\boldsymbol{c}} = \arg\max_{\boldsymbol{C}} p_M(\boldsymbol{c}|\boldsymbol{x})$, since it has shown the most competive behaviour in terms of acccuracy (Rodríguez et al., 2010).

Finally, we have to set the parameters of the multi-objective learning approach:

- Objective function: r repeated k-fold cross-validation evaluation method with $r = 10$ and $k = 10$ for each class variable.
- Individual size: 65.
- Size of the population: $P = 100\times$(individual size).
- Number of neighbours for each subproblem: $T = 20$.
- Stop criterion: $500,000$ evaluations (77 iterations of the algorithm).
- Number of replacements in the neighbourhood: $s = 2$.

In order to make all these experiments possible, we use different open source libraries in Java. For the classification utilities we use the DMLIB library and the Weka library (Witten and Frank, 2000) and for the multi-objective optimization utilities we use the jMetal library (Durillo et al., 2010).

7.3.4 Results

The proposed approach returns a set of non-dominated multi-dimensional classifiers. It shows different trade-off solutions to the multi-dimensional classification problem with different accuracy values for the different class variables.

The proposed learning approach is subject to variability which comes from different sources. First, MOEA/D is not deterministic. Moreover, due to the k-fold cross-validation process used in the internal evaluation and the hold-out procedure used in the external evaluation, there is an additional source of variability (Rodríguez et al., 2010). So, a single execution of the algorithm cannot be considered representative. In order to show the behaviour of the algorithm, we have executed the algorithm 100 times with different MOEA/D seeds and different training and test sets ($\mathcal{S}^1$ and $\mathcal{S}^2$) for the hold-out procedure (note that this procedure can be understood as a 100 times repeated hold-out estimator). Then we have plotted all the obtained Pareto fronts in a single figure. The accuracies have been estimated using the same pairs training-test for all the classifiers considered.

First, we compare our multi-objective learning approach with the multi-dimensional classifiers (MDnB and MDTAN) proposed by de Waal and van der Gaag (2007) and van der Gaag and de Waal (2006) (see Figure 7.4). We also compare our approach with the one-class approaches to multi-dimensional classification mentioned in Section 2.2.2. For that purpose, we have chosen naïve Bayes classifier (nB) (Langley et al., 1992; Minsky, 1961), tree-augmented Bayesian classifier (TAN) (Friedman et al., 1997), support vector machines (SVM) (Vapnik, 1995; Cristianini, 2010; Burges, 1998) and two Instance-based learning classifiers (KNN with $K = 1$ and $K = 3$) (Aha et al., 1991). In order to make the experiments replicable, we have used

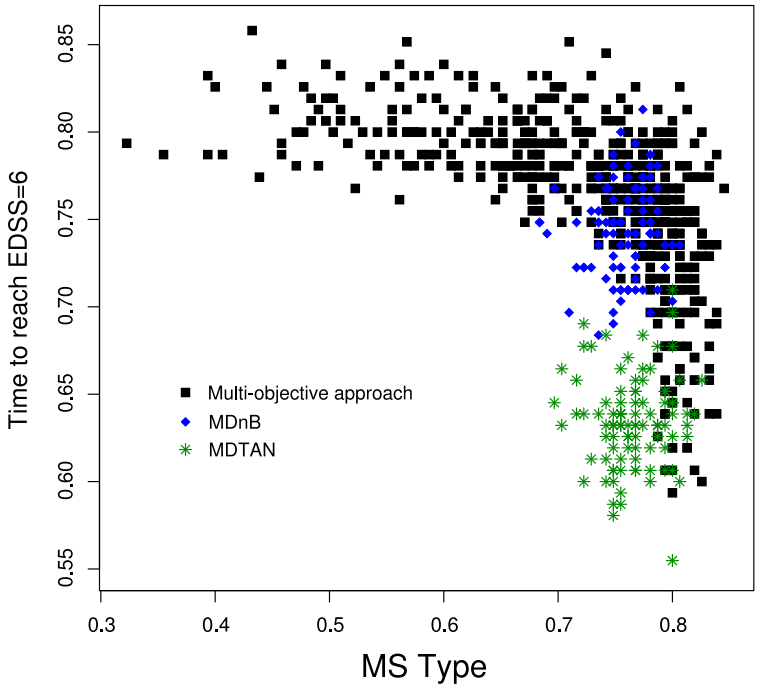


Fig. 7.4. 100 executions of the multi-objective learning approach versus 100 MDnB and MDTAN executions.

the SVM (weka.classifiers.functions.SMO) and kNN (weka.classifiers.lazy.IBk) classifiers provided by weka (Witten and Frank, 2000) with their default parameters. We also compare the proposed approach versus the use of a compound class variable and versus the use of multiple classifiers [2] (see Figures 7.5 and 7.6 respectively).

The best accuracy values of the classifiers in the Pareto fronts are between 80% and 85% for each class variable, and they take up an extensive part of the objective function space, so the physician, as he/she has more classifiers to compare, can select more suitable classifiers. Figure 7.4 illustrates that the classifiers obtained with the proposed approach dominate the MDTAN and MDnB classifiers. The multi-dimensional version of TAN, (MDTAN), shows competitive results for the *subtype of MS* class variable but, unfortunately, it shows a bad behaviour for the *time to reach* $EDSS = 6$ class variable.

Figure 7.5 shows the results of the proposed approach versus nB, TAN, SVM, 1NN and 3NN, all of them with the compound class variable (i.e. the

[2] Note that in the multiple classifiers approach, each dot representing nB, TAN, SVM, 1NN and 3NN in Figure 7.6 is, in fact, 2 classifiers: one for each class variable.

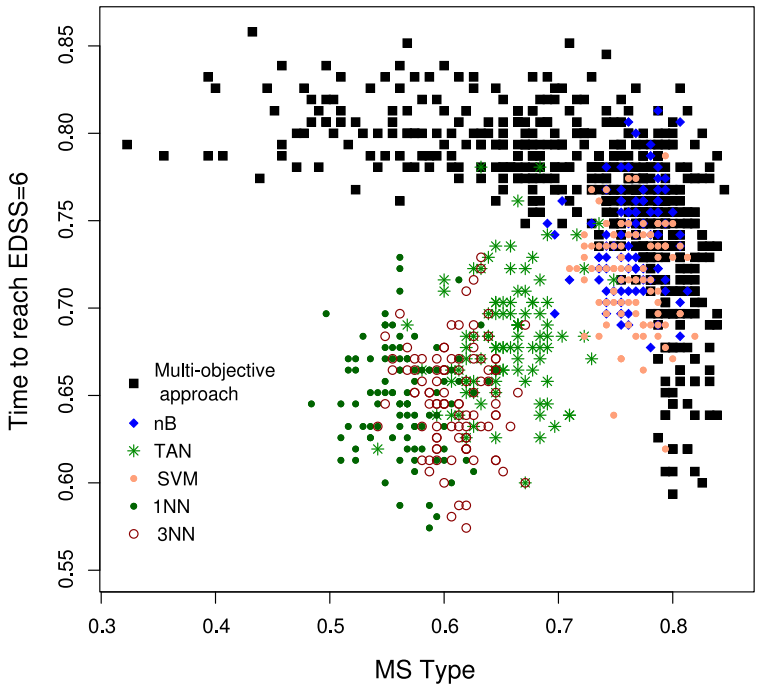


Fig. 7.5. 100 executions of the multi-objective learning approach versus 100 compound class variable executions using nB, TAN, SVM, 1NN and 3NN.

Cartesian product of both class variables). The classifiers in the Pareto Front dominate all the TAN, 1NN and 3NN classifiers and most of the SVM and nB classifiers.

Figure 7.6 shows the results of our approach versus multiple classifiers (one for each class variable) based on nB, TAN, SVM, and kNN with $k = 1, 3$. nB, TAN and kNN show a similar behaviour compared to their compound class approaches. In contrast, multiple SVM classifiers show a notably competitive behaviour, with solutions that are non-dominated with the solutions obtained by our approach. However, this approach lacks the interpretability of multi-dimensional Bayesian classifiers.

Another interesting point is that, in this particular data set and in both approaches (compund class variable and multiple classifiers), the classifiers based on a naive Bayes structure (2 nB, a single nB with a compound class variable and MDnB) seem to have a better behaviour than the classifiers based on TAN structures (2 TAN, a single TAN with a compound class variable and MDTAN). This is not an unexpected result, as the naive Bayes classifier is known to have successful results in medical domains (Kononenko, 1990; Ohmann et al., 1996; Mani et al., 1997; Movellan et al., 2002). However, the use of SVM classifiers (2 SVM and a single SVM with a compound class variable)

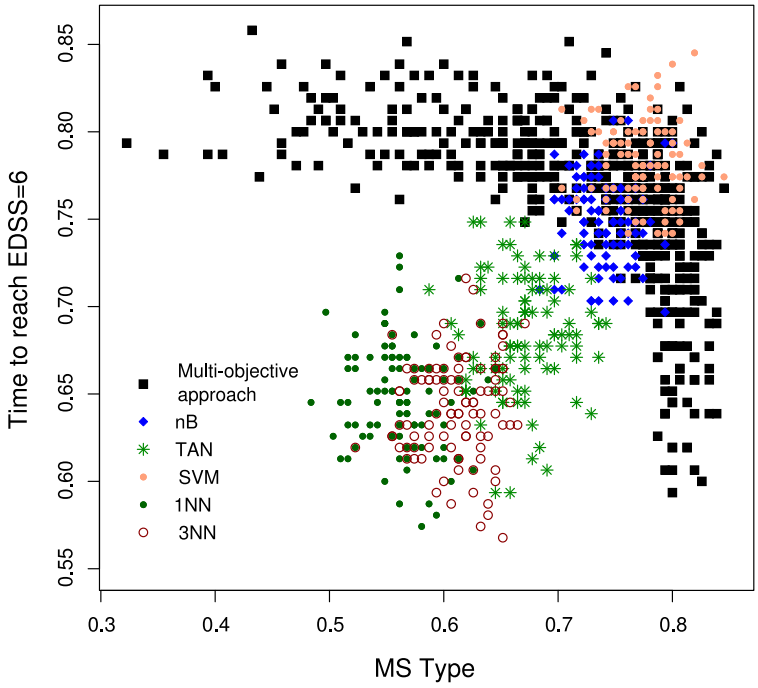


Fig. 7.6. 100 executions of the multi-objective learning approach versus 100 multiple classifiers executions using nB, TAN, SVM, 1NN and 3NN.

shows the best behavior in terms of accuracy. Finally, in both approaches the instance-based classifiers (1NN and 3NN), have the worst behaviour.

7.4 Conclusions

Nowadays, multiple sclerosis is one of the most common neurological diseases in young adults. There are different treatment options, with different efficacy and side effects depending on the type and disability severity status of the disease. In this article, we would like to assist a physician to identify the expected progression of the disease, so the most suitable treatment for each particular patient can be chosen. To that end, we developed an application that tries to identify the expected progression of multiple sclerosis in patients that have been recently diagnosed.

In order to develop the application, we set out a predictive model that, given genetic and environmental factors as predictive variables, is supposed to predict some variables that define the progression of the disease. The selected class variables are *subtype of the disease* and the *time to reach* $EDSS = 6$, as they can help a physician to predict the progression of the disease. As

there is more than one class variable and they seem to be correlated, we propose to approach the problem as a multi-dimensional classification problem and construct multi-dimensional Bayesian network classifiers because they can take into account the probabilistic relations among the class variables. We present a novel multi-objective learning of multi-dimensional J/K dependences Bayesian classifier (with $J = 1$ and $K = 2$). The use of a multi-objective learning approach allows to maximize the accuracy of all the class variables simultaneously. It returns a set of Bayesian classifiers which allows the physician to select the most suitable in terms of accuracy or based on the medical expert knowledge about MS.

The results show that there are several classifiers in the obtained Pareto fronts, with high accuracy values for both class variables (between 80% and 85%), that dominate the solutions obtained for MDnB, MDTAN, nB, TAN, SVM and kNN with $k = 1, 3$ for both one-dimensional approaches (compound class variable and multiple classifiers). However, some of the SVM classifiers for the multiple classifiers approach are non-dominated with the best classifiers in our approach. This approach produces accurated classifiers but with a low interpretability of the model.

Part V

Conclusions and Future Work

8

Conclusions

In this chapter, we present the general conclusions of this dissertation. More specific conclusions have been exposed in each corresponding chapter. We also present the publications produced during the thesis and the future paths to extend the presented work.

The topics presented in this work are error estimation on supervised classification and multi-dimensional supervised classification. The main contributions are focused on analyzing the statistical properties of the most popular classification error estimators and the learning of multi-dimensional Bayesian network classifiers.

The rest of the chapter is organized as follows: Section 8.1 summarises the contributions of this dissertation. In Section 8.2 we provide the list of papers published during this Ph.D. Finally, in Section 8.3 we expose some future work lines.

8.1 Contributions

The contributions of this dissertation can be divided into two main categories: contributions related with error estimation on supervised classification and contributions related to multi-dimensional classification.

8.1.1 Contributions on classification error estimation

This part of the dissertation is devoted to the analysis of the statistical properties (bias and variance) of the most popular classification error estimators used on supervised classification (resubstitution (Devroye and Wagner, 1979), holdout (Larson, 1931), repeated holdout, k-fold cross-validation (Stone, 1974), repeated k-fold cross validation (Kohavi, 1995a), simple bootstrap and 0.632 bootstrap (Efron and Tibshirani, 1993)) with and without stratification. For that issue, we present a unified framework to analyze the decomposition of

the variance of different error estimators considering the nature of the variance (irreducible/reducible variance) and the different sources of sensitivity (internal/external sensitivity). By means of the proposed framework, we can quantify the contributions of the irreducible variance, the internal sensitivity and the external sensitivity to the total variance for the considered error estimators.

The motivation of this analysis is the importance of the statistical properties of classification error estimators, not only to predict the future error of a classifier, but also for model selection. If the goal is to estimate the classification error of a particular classifier, the desired estimator should have low bias and low variance. However, if the goal is the model selection, in order to make fair comparisons the chosen estimator should have low variance assuming that the bias term is independent of the considered classifiers.

We have empirically studied the relative importance of the different sources of the variances for the error estimators considered. First, we have observed that in the performed experimentation the external sensitivity is the main part of the variance for all the error estimators considered, whereas the internal sensitivity is a insignificant portion of the whole variance. Besides, we have compared the variance and the Absolute Bias Differences (ABDs) among the different error estimators. The comparison in terms of variance allows to select the most stable error estimators whereas the comparison among the bias differences allows to select the most fair error estimator for model selection.

Based on the obtained results, we have proposed a set of practical recommendations in order to select the most appropriate error estimator for model selection taking into account the computational complexity: both 0.632 bootstrap with $B = 1000$ or stratified repeated 5-fold cross-validation estimators for induction algorithms with low computational complexity, or stratified 5-fold cross validation for induction algorithms with a high computational complexity.

8.1.2 Contributions on multi-dimensional classification

The second contribution of this dissertation is related with multi-dimensional classification. First, we present a learning approach of multi-dimensional Bayesian network classifiers and then, we present an application in order to help a physician to identify the expected progression of multiple sclerosis and plan the most suitable treatment.

8.1.2.1 Learning multi-dimensional Bayesian network classifiers by means of a multi-objective approach

In Section 2.2.2 we presented multi-dimensional supervised classification as an extension of the classical supervised classification task to the prediction of multiple class variables. There are some one-dimensional approaches to deal

with this problem, but they do not model the real multi-dimensional idiosyncracy of the problem or generate a huge number of parameters for a high number of class variables. In order to exploit the correlations among the class variables, van der Gaag and de Waal (2006); de Waal and van der Gaag (2007) presented multi-dimensional Bayesian network classifiers (MDBCs). Recently, the machine learning community has developed some learning approaches of MDBCs (van der Gaag and de Waal, 2006; de Waal and van der Gaag, 2007; Zaragoza et al., 2011a; Borchani et al., 2010; Bielza et al., 2011; Ortigosa-Hernández et al., 2012).

In Chapter 6, we present a learning approach for MDBCs following a multi-objective strategy which considers the accuracy of each class variable separately as the functions to optimize. The solution of the learning approach is a Pareto set of non-dominated multi-dimensional Bayesian network classifiers and their accuracies for the different class variables, so a decision maker can easily choose by hand the classifier that best suits the particular problem and domain.

The proposed learning approach has been compared with the main alternatives to deal with multi-dimensional classification in artificial and real-world domains. The classifiers obtained with the proposed learning approach come up to the best mean accuracy and single accuracy for all the class variables, and it seems to improve its performance as the number of class variables increases. We can conclude that the proposed approach has a competitive behaviour, especially when the number of class variables involved is high.

8.1.2.2 Application of multi-dimensional Bayesian network classifiers on medicine domains

Once we have proposed a learning approach of MDBCs, it should be interesting to test it in any practical domain. To that end, in Chapter 7 we develop an application that tries to identify the expected progression of multiple sclerosis in patients that have been recently diagnosed. The data set used to apply the learning approach presented in Chapter 6 is composed of DNA and clinical information of patients from the MS Center at the VU University Medical Center (VUmc) in Amsterdam.

In Multiple Sclerosis there are different treatment options available with different efficacy and side effects, depending on the type and progression (time to reach certain severity level) of the disease. Given that both variables are correlated, we consider that the knowledge gained across multi-dimensional classification methods could help to make the strategy for disease management more suitable.

The multi-dimensional Bayesian network classifier used in this application is the *multi-dimensional J/K dependences Bayesian classifier* (Rodríguez and Lozano, 2010) with $J = 1$ and $K = 2$. For that issue, we have slighty modified the multi-objective approach presented in Chapter 6. It returns a Pareto set

of classifiers with different accuracy values for each of the classes. A physician should select the most suitable in terms of accuracy or based on the medical expert knowledge about MS.

The results show that several classifiers in the obtained Pareto fronts dominate the solutions obtained for MDnB, MDTAN, nB, TAN, SVM and kNN with $k = 1, 3$ for both one-dimensional approaches (compound class variable and multiple classifiers). However, some of the SVM classifiers for the multiple classifiers approach are non-dominated with the best classifiers in our approach. The difference is that even though SVM classifiers reach accurate classifiers they have a low interpretability of the model.

8.2 Publications

In this section, we present the publications and submissions produced during this thesis. First we present the publications directly derived from the thesis and then we present the colaborations with other researchers relative to this thesis.

8.2.1 Publications of this thesis

A. Refereed JCR journals

- **Rodríguez, J. D.**, Pérez, A. & Lozano, J. A. (2013). Learning Bayesian network classifiers for multi-dimensional supervised classification problems by means of a multi-objective approach. *Knowledge and Information Systems.* Submitted
- **Rodríguez, J. D.**, Pérez, A. & Lozano, J. A. (2013). A General Framework for the Statistical Analysis of the Sources of Variance for Classification Error Estimators. *Pattern Recognition*, 46(3), 855-864.
- **Rodríguez, J. D.**, Pérez, A., Arteta, D., Tejedor, D. & Lozano, J. A. (2012). Using Multi-Dimensional Bayesian Network Classifiers to Assist the Treatment of Multiple Sclerosis. *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, 42(6), 1705-1715.
- **Rodríguez, J. D.**, Pérez, A. & Lozano, J. A. (2010). Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32, 569-575.

B. Conference communications

- **Rodríguez, J. D.** & Lozano, J. A (2008). Multi-objective learning of multi-dimensional Bayesian classifiers. *In Proceedings of the Eighth International Conference on Hybrid Intelligent Systems*, pages 501-506.

- **Rodríguez, J. D.** & Lozano, J. A (2007). Repeated stratified k-fold cross-validation on supervised classification with naive Bayes classifier: An empirical analysis. *In Proceedings of the Fifth Spanish Workshop on Data Mining and Learning, (TAMIDA 2007)*, pages 65 -74

C. Technical reports

- **Rodríguez, J. D.** & Lozano, J. A. (2010). Learning Bayesian network classifiers for multidimensional supervised classification problems by means of a multiobjective approach, *UPV/EHU Technical Report.*
- **Rodríguez, J. D.**, Pérez, A. & Lozano, J. A. (2009). A sensitivity study of bias and variance of k-fold cross-validation in prediction error estimation, *UPV/EHU Technical Report.*

8.2.2 Colaborations with other researchers relative to this thesis

A. Refereed JCR journals

- Fernandes, J. A., Lozano, J. A., Inza, I., Irigoien, X., Pérez, A. & **Rodríguez, J. D.** (2013). Supervised pre-processing approaches in multiple class variables classification for fish recruirment forecasting. *Environmental Modelling & Software*, 40, 245-254.
- Ortigosa-Hernández, J., **Rodríguez, J. D.**, Alzate, L., Lucania, M., Inza, I. & Lozano, J. A. (2012). Approaching Sentiment Analysis by Using Semi-supervised Learning of Multi-dimensional Classifiers. *Neurocomputing*, 92, 98-115.

B. Conference communications

- Ortigosa-Hernández, J., **Rodríguez, J. D.**, Alzate, L., Inza, I. & Lozano, J. A (2010). A Semi-supervised Approach to Multi-dimensional Classification with Application to Sentiment Analysis. *In Trocoso, A. & Requelme, J. C. (editors), Actas del V Simposio de Teoría y Aplicaciones de Minería de Datos (TAMIDA 2010)*, pages 129-138.

C. Technical reports

- Ortigosa-Hernández, J., **Rodríguez, J. D.**, Alzate, L., Lucania, M., Inza, I. & Lozano, J. A. (2011). Approaching Sentiment Analysis by Using Semi-supervised Learning of Multi-dimensional Classifiers, *UPV/EHU Technical Report.*

8.3 Future work

Undoubtedly, there is a lot of research to be done regarding the areas discussed in this work. Next, we will extend the future work presented by this thesis in error estimation on supervised classification and multi-dimensional classification.

8.3.1 Error estimation on supervised classification

In this work, we presented a framework for the analysis of the statistical properties of classification error estimators on supervised classification. The presented framework uses accuracy as the score, but we consider that it could be an interesting issue to adapt the proposed framework to other scores (for example AUC, precision, sensitivity...).

The study presented has been developed in a wide domain of artificial data sets in order to obtain robust conclusions, as the use of data sets obtained from artificial domains allows to compute exactly the implied quantities. So, the conclusions could be globally applied to real data sets. However, we think that the proposed framework can be applied with guarantees in data sets from the real world if they are large enough. The reason is that in real world data sets the underlying distribution is usually unknown, so it should be estimated from data in order to perform an approximate analysis of the sources of the variance for the different estimators. It is known that the quality of the estimation of the underlying distribution depends on the size of the data set and thus, we consider that the framework could be applied with guarantees in large data sets from the real world. This application will give interesting information about the used real-world data set that could help in the classification task.

Finally, we have observed that in the performed experimentation the external sensitivity is the main part of the variance for all the error estimators considered, whereas for some of the estimators the internal sensitivity is a insignificant portion of the whole variance (repeated versions reduce this part of the variance). This suggests that in the future we should concentrate our efforts on designing error estimators which try to reduce the external sensitivity.

8.3.2 Multi-dimensional classification

Multi-dimensional classification is a relatively new field, so there is room for many new developments. First, there are some paths where it is worth extending the proposed multi-objective learning approach. As it returns a set of classifiers, it would be interesting to develop techniques to help a decision maker to choose an appropriate classifier from the Pareto front. Moreover, the multi-objective learning approach could be led by different scores instead of the single accuracy of each class variable. For example, we can use sensitivity or specificity for each class variable as the functions to optimize. Finally, the external evaluation of the Pareto front can be refined, as we use a hold-out procedure that is penalized by a high variance.

In this dissertation we have focused on learning MDBCs with a multi-objective learning approach. However, there are several ways to learn MDBCs (see Section 5.4), so it will be interesting to develop new learning approaches using different paradigms. For instance, we can try to learn MDBCs by means of Estimation Distribution Algorithms (EDAs) using the joint accuracy as

the score. Continuing on a different vein, it could be also interesting to use multi-objective optimization techniques oriented to feature selection in multi-dimensional classification.

Finally, it will be interesting to develop more real-world applications of MDBCs. If we focus on the proposed Multiple Sclerosis application, we would like to work on the detailed prediction of the *time to reach* $EDSS = 6$. It is of practical interest to give an uncertainty measure of the classifiers to the physician, so he/she can make a more suitable decision. We could give not only the prediction of the *time to reach* $EDSS = 6$, but also the probability for each of the possible values that the variable can take. For example, with the current approach we classify a specific individual with a value for *time to reach* $EDSS = 6$ (for example that the patient is going to reach level 6 between 5 and 10 years after being diagnosed), but we do not give the certainty of the classification. We could extend this information giving the probability of each of the different possible values. For example:

- Reach level 6 between 5 and 10 years: 30%
- Reach level 6 between 10 and 20 years: 50%
- Reach level 6 in more than 20 years: 20%

Moreover, we would like to define, using the physicians' expert knowledge, cost matrices in order to obtain the most suitable classifiers, as they would take into account the weight of the different classification errors.

References

Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Aliferis, C., Tsamardinos, I., and Statnikov, A. (2003). HITON, a novel markov blanket algorithm for optimal variable selection. In *AMIA Annual Symposium*, pages 21–25.

Anscombe, F. J. (1967). Topics in the investigation of linear relations fitted by the method of least squares. *Journal of the Royal Statistical Society Series B*, 29:1–52.

Asuncion, A. and Newman, D. J. (2007). UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Bakir, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., and Vishwanathan, S. V. N. (2007). *Predicting Structured Data (Neural Information Processing)*. The MIT Press.

Bengio, Y. and Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. *The Journal of Machine Learning Research*, 5:1089–1105.

Bengio, Y. and Grandvalet, Y. (2005). *Bias in Estimating the Variance of K-Fold Cross-Validation*, volume 1 of *Statistical Modeling and Analysis for Complex Data Problem*. Springer.

Bielza, C., Li, G., and Larrañaga, P. (2011). Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blanco, R. (2005). *Learning Bayesian network from data with factorisation and classification purposes. Applications in Biomedicine*. PhD thesis, University of the Basque Country.

Borchani, H., Bielza, C., and Larrañaga, P. (2010). Learning cb-decomposable multi-dimensional bayesian network classifiers. In *Proceedings of the 5th*

European Workshop on Probabilistic Graphical Models (PGM10), pages 25–32.

Borchani, H., Bielza, C., and Larrañaga, P. (2011). Learning multi-dimensional bayesian network classifiers using markov blankets: A case study in the prediction of hiv protease inhibitors. In *AIME'11 Workshop on Probabilistic Problem Solving in Biomedicine*.

Braga-Neto, U., Hashimoto, R., Dougherty, E., Nguyen, D., and Carroll, R. (2004). Is cross-validation better than resubstitution for ranking genes? *Bioinformatics*, 20(2):253–258.

Braga-Neto, U. M. (2005). Small-sample error estimation: mythology versus mathematics. In *Proceedings of SPIE (2005)*, volume 5916, pages 304–314.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth.

Burges, C. (1998). An introduction to support vector machines and other kernel-based learning methods. *Data Mining and Knowledge Discovery*, (2):121–167.

Burman, P. (1989). A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika*, 76:503–514.

Burman, P., Chow, E., and Nolan, D. (1994). A cross-validation method for dependent data. *Biometrika*, 81:351–358.

Caruana, R. (1997). Multi-task learning. *Machine Learning*, 28:41–75.

Castillo, E., Gutiérrez, J., and Hadi, A. (1997). *Expert systems and probabilistic network models*. Monographs in Computer Science. Springer.

Cesa-Bianchi, N., Gentile, C., and Zaniboni, L. (2006). Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research*, 7:31–54.

Cheng, W. and Hullermeier, E. (2009). Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225.

Chow, C. I., Member, S., and Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467.

Cochran, W. G. (1968). Commentary on estimation of error rates in discriminant analysis. *Technometrics*, 10:204–205.

Coello, C., Lamont, G., and Van Veldhuizen, D. (2006). *Evolutionary algorithms for solving multi-objective problems (genetic and evolutionary computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Compston, A. and Coles, A. (2002). Multiple sclerosis. *Lancet*, 359:1221–1231.

Cozman, F. (2000). Java Bayes: Bayesian networks in Java. http://www.cs.cmu.edu/ javabayes/home/index.html.

Cristianini, N. (2010). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press.

Daumé, H. and Marcu, D. (2005). Learning as search optimization: Approximate large margin methods for structured prediction. In *International Conference in Machine Learning (ICML)*, pages 169–176.

Dawid, A. (1979). Conditional independence in statistical theory. *Journal of royal statistics Society*, 41:1–3.

Dawid, A. P. (1992). Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2(1):25–36.

de Waal, P. and van der Gaag, L. (2007). Inference and learning in multi-dimensional Bayesian network classifiers. *Lecture Notes in Artificial Intelligence*, 4724:501–511.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

Devroye, L. and Wagner, T. (1979). Distribution-free perfomance bounds with the resubstitution error estimate. *IEEE Transactions on Information Theory*, 25(2):208–210.

Dos Santos, E. M., Sabourin, R., and Maupin, P. (2009). Overfitting cautious selection of classifier ensembles with genetic algorithms. *Inf. Fusion*, 10(2):150–162.

Dougherty, J., Kohavi, R., and Sahami, M. (1995). *Supervised and unsupervised discretization of continuous features*, volume 54, pages 194–202. Morgan Kaufmann Publishers Inc.

Droge, B. (1996). *Some comments on cross-validation*, volume 1 of *Statistical theory and computational aspects of smoothing*.

Duda, R., Hart, P., and Stork, D. (2001). *Pattern classification*. Wiley Interscience.

Dumais, S. and Chen, H. (2000). Hierarchical classification of web content. In *SIGIR '00: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 256–263, New York, NY, USA. ACM.

Durillo, J., Nebro, A., and Alba, E. (2010). The jmetal framework for multi-objective optimization: Design and architecture. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –8.

Durillo, J., Nebro, A., Luna, F., Dorronsoro, B., and Alba, E. (2006). jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. Technical Report ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos.

Efron, B. and Tibshirani, R. (1993). *An introduction to the Bootstrap*, volume 57 of *Monographs on statistics and applied probability*. Chapman and Hall.

Efron, B. and Tibshirani, R. (1995). Cross-validation and the Bootstrap: Estimating the error rate of a prediction rule. Technical Report Technical Report 176, Dept. Statistics, Stanford University.

Fernandes, J. A., Lozano, J. A., Inza, I., Irigoien, X., Pérez, A., and Rodríguez, J. D. (2013). Supervised pre-processing approaches in multiple class-

variables classification for fish recruitment forecasting. *Enviromental Modelling & software*, 40:245–254.

Friedman, N., Geiger, D., and Goldszmit, M. (1997). Bayesian network classifiers. *Machine Learning*, 29:131–163.

Friedman, N., Linial, M., Nachman, I., and Peer, D. (2000). Using Bayesian networks to analyze expression data. *Journal of computational biology a journal of computational molecular cell biology*, 7(3-4):601–620.

Frohman, E. M., Filippi, M., Stuve, O., Waxman, S. G., Corboy, J., Phillips, J. T., Lucchinetti, C., Wilken, J., Karandikar, N., Hemmer, B., Monson, N., De Keyser, J., Hartung, H., Steinman, L., Oksenberg, J. R., Cree, B. A. C., Hauser, S., and Racke, M. K. (2005). Characterizing the mechanisms of progression in multiple sclerosis: Evidence and new hypotheses for future directions. *Arch Neurol*, 62(9):1345–1356.

Fukunaga, K. and Hummels, D. M. (1989). Leave-one-out procedures for nonparametric error estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(4):421–423.

García, S. and Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694.

Hall, M. A. (1999). *Correlation-based feature selection for machine learning*. PhD thesis.

Hall, M. A. (2000). *Correlation-based feature selection for discrete and numeric class machine learning*, pages 359–366. Morgan Kaufmann Publishers Inc.

Hamerly, G. and Elkan, C. (2001). *Bayesian approaches to failure prediction for disk drives*, pages 202–209. Morgan Kaufmann Publishers Inc.

Handl, J., Kell, D., and Knowles, J. (2007). Multiobjective optimization in bioinformatics and computational biology. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):279–292.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The elements of statistical learning. Data mining, inference and prediction*. Springer series in Statistics. Springer-Verlag.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized additive models*. London: Chapman & Hall.

Hills, M. (1966). Allocation rules and their error rates. *Journal of the Royal Statistical Society Series B*, 28:1–31.

Horst, P. (1941). *Prediction of personal adjustment*, volume Bulletin 48. Social Science Research Council.

Hosmer, D. and Lemeshow, S. (1989). *Applied Logistic Regresion*. Jon Whiley and Sons.

Ide, J. S., Cozman, F. G., and Ramos, F. T. (2004). Generation of random bayesian networks with constraints on induced width. In *Proceedings of the 16th European Conference on Artificial Intelligence (2004)*, pages 323–327.

Jebara, T. (2004). *Machine Learning: Discriminative and Generative*. Kluwer Academic.

Jin, Y. J. Y. and Sendhoff, B. (2008). Pareto-based multiobjective machine learning: An overview and case studies. *IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews*, 38(3):397–415.

Kohavi, R. (1995a). A study of cross-validation and Bootstrap for accuracy estimation and model selection. pages 1137–1143. Morgan Kaufmann.

Kohavi, R. (1995b). *Wrappers for performance enhancement and oblivious decision graphs*. PhD thesis, Stanford University, Computer Science Department.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models. Principles and techniques*. The MIT Press, Cambridge, Massachusetts.

Kononenko, I. (1990). Current trends in knowledge acquisition, chapter comparision of inductive and naive Bayesian learning approaches to automatic knowledge acquisition.

Kullback, S. (1959). *Information theory and statistics*. John Wiley and Sons., New York.

Kurtzke, J. F. (1983). Rating neurologic impairment in multiple sclerosis: an expanded disability status scale (EDSS). *Neurology*, 33(11):1444–1452.

Lacave, C., Luque, M., and Díez, F. J. (2007). Explanation of Bayesian networks and influence diagrams in Elvira. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(4):952–965.

Lachenbruch, P. and Mickey, M. (1968). Estimation of error rates in discriminant analysis. *Technometrics*, 10:1–10.

Langley, P., Iba, W., and Thompson., K. (1992). An analysis of Bayesian classifiers. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 223–228.

Larrañaga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J. A., Armañanzas, R., Santafé, G., and Pérez, A. (2006). Machine learning in bioinformatics. *Briefings in Bioinformatics*, 17(1):86–112.

Larrañaga, P., Lozano, J. A., Peña, J. M., and Inza, I. (2005). Editorial. *Machine Learning*, 59(3):211–212.

Larson, S. C. (1931). The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology*, 22:45–55.

Lauritzen, S. L. (1996). *Graphical Models*. Oxford University Press.

Lauritzen, S. L., Dawid, A. P., Larsen, B. N., and Leimer, H. G. (1990). Independence properties of directed markov fields. *Networks*, 20(5):491–505.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society Series B Methodological*, 50(2):157–224.

Li, H. and Zhang, Q. (2009). Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):229–242.

Mani, S., Pazzani, M., and West, J. (1997). Knowledge discovery from a breast cancer database. In Keravnou, E., Garbay, C., Baud, R., and Wyatt, J.,

editors, *Artificial Intelligence in Medicine*, volume 1211 of *Lecture Notes in Computer Science*, pages 130–133. Springer Berlin / Heidelberg.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. *Dimension Contemporary German Arts And Letters*, 752(1):41–48.

McLachlan, G. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley.

Miettinen, K. (1999). *Nonlinear multiobjective optimization*. Kluwer Academic Publishers, Boston, MA.

Minsky, M. (1961). Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49(1):8–30.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill Education (ISE Editions).

Miyahara, K. and Pazzani, M. J. (2000). Collaborative filtering with the simple Bayesian classifier. *Matrix*, 8046(949):679 – 689.

Mosteller, F. and Tukey, J. W. (1968). Data analysis, including statistics. In Lindzey, G. and Aronson, E., editors, *Handbook of Social Psycholgy*, volume 2, chapter 10, pages 80–203. Addison-Wesley.

Movellan, J. R., Wachtler, T., Albright, T. D., and Sejnowski, T. (2002). Naive Bayesian coding of color in primary visual cortex. *Advances in Neural Information Processing Systems*, 15:221–228.

Nadeau, C. and Bengio, Y. (2003). Inference for the generalization error. *Machine Learning*, 52(3):239–281.

Neapolitan, R. (2003). *Learning Bayesian Networks*. Prentice Hall.

Ohmann, C., Moustakis, V., Yang, Q., and Lang, K. (1996). Evaluation of automatic knowledge acquisition techniques in the diagnosis of acute abdominal pain. acute abdominal pain study group. *Artificial Intelligence in Medicine*, 8(1):23–36.

Ortigosa-Hernández, J., Rodríguez, J. D., Alzate, L., Lucania, M., Inza, I., and Lozano, J. A. (2012). Approaching sentiment analysis by using semi-supervised learning of multi-dimensional classifiers. *Neurocomputing*, 92:98–115.

Osyczka, A. (1985). *Multicriteria optimization for engineering design*. Academic Press.

Pearl, J. (1988). *Probabilistic reasoning in intelligence systems*. Springer series in Statistics. Morgan-Kaufman.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.

Radtke, P. V. W., Wong, T., and Sabourin, R. (2009). Solution over-fit control in evolutionary multiobjective optimization of pattern classification systems. *IJPRAI*, 23(6):1107–1127.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359.

Rodríguez, J. D. and Lozano, J. A. (2010). Learning Bayesian network classifiers for multidimensional supervised classification problems by means of

a multiobjective approach. Technical Report EHU-KZAA-TR-3-2010, University of the Basque Country.

Rodriguez, J. D., Perez, A., Arteta, D., Tejedor, D., and Lozano, J. A. (2012). Using multidimensional bayesian network classifiers to assist the treatment of multiple sclerosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(6):1705 –1715.

Rodríguez, J. D., Pérez, A., and Lozano, J. A. (2010). Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):569–575.

Rodríguez, J. D., Pérez, A., and Lozano, J. A. (2013). An unified framework for the statistical analysis of the sources of variance for classification error estimators. *Pattern Recognition*, 46(3):855–864.

Rubinstein, Y. D. and Hastie, T. (1997). Discriminative vs informative learning. In *Proceedings of the third international conference on knowledge discovery and data mining*, pages 49–53. AAAI Press.

Sahami, M. (1996). Learning limited dependence Bayesian classifiers. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (1996)*, pages 335–338.

Santafé, G., Lozano, J. A., and Larrañaga, P. (2007). Discriminative vs. Generative Learning of Bayesian Network Classifiers. In *ECSQARU*, pages 453–464.

Schaffer, C. (1993). Selecting a classification method by cross-validation. *Machine Learning*, 13:135–143. 10.1007/BF00993106.

Sombekke, M. H., Arteta, D., van de Wiel, M. A., Crusius, J. B. A., Tejedor, D., Killestein, J., Martínez, A., Pea, A. S., Polman, C. H., and Uitdehaag, B. M. (2010). Analysis of multiple candidate genes in association with phenotypes of multiple sclerosis. *Multiple Sclerosis*, 16(6):652–659.

Stadler, W. (1988). *Multicriteria optimization in engineering and in the sciences*. Springer.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society Series B*, 36:111–147.

Tax, D. M. J. and Duin, R. P. W. (2002). Using two-class classifiers for multiclass classification. *International Conference on Pattern Recognition*, 2:1051–4651.

Trohidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. (2008). Multilabel classification of music into emotions. In *Proceedings of the 2008 International Conference on Music Information Retrieval (ISMIR 2008)*, pages 325–330.

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., and Vlahavas, I. (2011). Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414.

van der Gaag, L. and de Waal, P. (2006). Muti-dimensional Bayesian network classifiers. In *Proceedings of the Third European Workshop in Probabilistic Graphical Models*, pages 107–114.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

Whittaker, J. (1991). *Graphical Models in Applied Multivariate Statistics*. Series in Probability and Mathematical Statistics. Wiley.

Witten, I. and Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with Java implementations*. The Morgan Kaufmann series in Data Management Systems.

Zaragoza, J. C., Sucar, L. E., and Morales, E. F. (2011a). A two-step method to learn multidimensional bayesian network classifiers based on mutual information measures. In *FLAIRS Conference*.

Zaragoza, J. C., Sucar, L. E., Morales, E. F., Bielza, C., and Larrañaga, P. (2011b). Bayesian chain classifiers for multidimensional classification. In *IJCAI*, pages 2192–2197.

Zhang, M.-L. and Zhou, Z.-H. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

Zhang, Q. and Li, H. (2007). MOEA/D: A multi-objective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–7314. Winner of the IEEE Transactions on Evolutionary Computation Outstanding Paper Award.