UNIVERSITY OF CASTILLA-LA MANCHA
Computing Systems Department

New models and algorithms for semi-naive Bayesian classification focused on the AODE paradigm

Ana María Martínez Fernández

Ph.D. Thesis, June 2012

UNIVERSITY OF CASTILLA-LA MANCHA
Computing Systems Department

PhD. Program:
Doctorado en Tecnologías Informáticas Avanzadas

PhD. Thesis Dissertation:

New models and algorithms for semi-naive Bayesian classification focused on the AODE paradigm

PhD. Student:
Ana María Martínez Fernández

Advisors:
José A. Gámez and M. Julia Flores

To my parents
(A mis padres)

Acknowledgements

I would like to start by acknowledging to all those who, for different reasons, have made this 4-year amazing experience possible. I find it highly rewarding on a professional level, but also on the personal level.

This thesis would not be what it is without the impeccable supervision of José A. Gámez and M. Julia Flores. José A. has just been essential from the very beginning. I admire his brilliant ideas and his capability of helping me out of any quagmire. Almost magically, everything becomes much simpler after his explanations. Julia has been the best company, advisor and friend along the way. She has made these four years go nicely and pleasantly. Thanks to both of them for their comprehension and their trust, for giving me this opportunity.

I consider José Miguel Puerta to be the third unofficial supervisor of this thesis. I have been lucky to count on his great intuition, specially in the first part of this thesis.

I could not imagine what it would be like research on a thesis, without a group of other students with whom to share, not only scientific enquiries, but also the daily routines. Thanks to all the SIMD members, ex-members, visitors and undergrad students who have been around the lab, making the working time much more pleasant. Specially to my lab mates throughout this time: Pablo, Juan Luis, Jens, Juanin, Alex, Jesús, Benja, Maite and Cózar. Not to forget my neighbours from RETICS, specially María, last minute "trade", but not least.

I would also like to thank Thomas D. Nielsen for hosting me in the Maskinintelligens group. It was a great stay, where I learnt a lot, specially from him, from his knowledge and his work methodology. I was very fortunate to work and learn also from Helge Langseth and Shengtong Zhong.

Thanks to Antonio Salmerón, his warm welcome in Almería and his wise advice in terms of MTEs. I take the opportunity to acknowledge all the members from the joint projects with the university of Almería, Granada and Madrid (UPM), for all the enlightening meetings along these years.

My trip to Melbourne was enriching in every sense. I had the opportunity to work under the supervision of Geoff Webb, clear motivator of this thesis. I have to acknowledge him for hosting me at the faculty of information technology, as part of the KnowSysLab. His joint work has inspired a great part of this dissertation, working with him has been a real honour. Thanks to Ann Nicholson as well, for all her help and valuable company during my stay.

I want to acknowledge all the colleagues and friends met in these trips, who have helped to make it all worth while. Special mention to Carlos Contreras, so wise and supportive, a great friend.

I would like to show my gratitude to the anonymous and non anonymous reviewers of the content included in this thesis report. Their comments have undoubtedly helped to improve its quality.

Last but sure not least, I want to show endless gratitude to my family. First of all my parents, Gloria and Julio, I could write several pages expressing my appreciation, they would deserve every word. They have given me the courage to continue, to work calmly but steady, to enjoy the process. Many thanks to my sisters, Emi and Gloria, for being always there helping and supporting on everything, their invaluable sincerity; and also to my grandparents. To Ernesto, for making me wonder the real value of Bayesian networks regularly. To my niece, Julia, who has made this last 10 months to have a new meaning. Also to Charo, Vito and Pili, for all the time they have saved me this last year.

And of course to David, for his understanding, his patience and his support all this time. He has given me the positive energy and the courage I needed to complete this stage, "thank you" sounds tiny here.

This work has been financially supported by the FPU grant with reference number AP2007-02736. Also, by the Consejería de Educación y Ciencia (JCCM) under Project PCI08-0048-8577574 and the Spanish Ministerio de Educación y Tecnología under Projects TIN2007-67418-C03-01 and TIN2010-20900-C04-02,03.

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Introduction

Chapter 1

Introduction

Prediction is very difficult, especially about the future.
Niels Bohr. (1885 - 1962)
Danish physicist

1.1 Motivation

In everyday life we encounter a large number of classification problems. Many of them are solved intuitively, i.e. without an explicit method or algorithm, such as: what type of person is "this"?, which is the gender of this film?, where do I place (classify) this new dish I tested today? (with all its subtypes). We generally carry out these classification processes subconsciously, based on previous experience. This "previous experience" translates into training data in machine learning. The main objective then is to recognize complex patterns on these data in order to make intelligent decisions.

The term *database* is defined as "a structured set of data held in a computer, especially one that is accessible in various ways" according to the Oxford English dictionary[1]. With the permanent increase of hardware capabilities and improvement of network technologies, the size and performance of databases have steady grown at exponential rate. Although the task of data mining to provide a clean and proper database is not trivial, in this the-

[1] http://www.oed.com/

sis we assume we have this available to perform the classification task, except for the need to pre-process the databases with filters for discretization, treatment of missing values or useless attributes. For our purposes, "only" the content of the database organised in m instances with n attributes per instance plus the class (which must be discrete)[1] is needed. This is generally called a *dataset*, a term we use from now on to refer to the group of instances (also referred to as examples or records) treated in the classification process.

Nevertheless, these data may have different characteristics to consider prior to the application of a particular classification algorithm. The alternatives are massive and depend on the problem to deal with.

There are multiple paradigms for classification to begin with, such as: Bayesian networks (BNs), decision trees, rule induction, artificial neural networks, genetic programming, support vector machines, etc. In this thesis we focus on the first of these, which might be seen as a combination of statistical techniques and graphical models. BNs provide several advantages to the classification task. The most important is the fact that the networks store information about existing dependencies among the variables involved. This makes them capable of providing a visual representation of the relationships between variables, and at the same time to deal with uncertainty, very frequently present in real world.

In this dissertation we particularly focus on efficient Bayesian network classifiers (BNCs), that either do not perform structural learning or it is very simple. Learning the structure of a network can take a long time and effort, especially in the case of datasets of high dimensionality. That is why it is often convenient to consider a partially or totally pre-fixed structure from which the conditional probability tables are learnt. The most simple of these structures is the one used by the naive Bayes (NB) classifier [Duda & Hart, 1973], which assumes all the attributes are independent given the class. In spite of its naive assumption, it performs surprisingly well in certain domains. Hence, numerous techniques have been proposed to improve the accuracy of NB by alleviating the attribute interdependence problem. We refer to

[1]Note that we are referring here to a finite set of labels, if the class to predict is of a continuous type it becomes a problem of *regression*, which is out of the scope of this work.

them as semi-naive BNCs, a term introduced by Kononenko [1991]. Among these proposals, the Averaged One-Dependence Classifier (AODE) [Webb *et al.*, 2005] has proved to be significantly better in terms of error reduction compared to many other semi-naive techniques, maintaining under control its time and space complexity in training and classification time [Zheng & Webb, 2005].

For the particular case of AODE, computational needs in terms of memory storage and classification time (which is quadratic in the number of attributes) may be too large for some databases. In this thesis a new classifier (derived from AODE) is proposed to tackle some of these inconvenients.

Since the natural domain of AODE and many others BNCs consists on discrete variables, we find interesting to study the different alternatives for handling numeric attributes, commonly present in real databases. Among these options the discretization pre-processing step and the use of distribution functions, such as the Gaussian distribution, or mixtures of truncated exponentials (MTEs), will be studied in depth in Section 2.2 and Chapters 4 to 7 of this dissertation.

We have noticed that all the alternatives we propose may be beneficial for a large group of databases, but not for all. Hence, we consider to find out in advance, according to the characteristics of a particular dataset, which classifier among the BNCs is the most suitable in each case. To this purpose we study the domain of competence for several semi-naive classifiers according to different complexity measures [Ho & Basu, 2002].

1.2 Organisation of the dissertation

This dissertation is structured in five parts.

Part I comprises Chapters 1 and 2. The first chapter includes the introduction and draws the structure of the thesis (that you are reading at the moment). The second chapter includes three sections with information of the state-of-the-art on the three main subtopics that form this thesis: 1) overview in supervised classification and literature review of semi-naive BNCs; 2) options to handle continuous variables when applying BNCs; and finally, 3) complexity measures for determining the domain of competence of a classifier.

Parts II, III and IV include the following contributions of the thesis:

- Part II is divided in Chapters 3, 4 and 5, and contains the description and results of new classifiers, derived from AODE to overcome its limitations. First of all, Chapter 3 presents the Hidden One-Dependence Classifier (HODE). This classifier estimates a new mixture variable with the aim of capturing the significant interdependencies among variables included in AODE throughout the different models, so that both space needs and classification time are reduced. Chapter 4 introduces GAODE and HAODE, two classifiers designed to handle numeric variables in a more direct way than AODE. For this purpose, GAODE uses conditional Gaussian networks (CGNs) whereas HAODE discretizes a numeric attribute exclusively when it plays the role of superparent. In order to avoid the Gaussian assumption, in Chapter 5, we resort to more general probability distributions, in particular, the use of Mixture of Truncated Exponentials (MTEs) through the proposal of the MTE-AODE classifier.

- Part III is divided in Chapters 6 and 7, and deals with discretization techniques in semi-naive BNCs. Chapter 6 shows an study on the effect of different traditional supervised and non-supervised discretization techniques on NB, AODE, tree augmented naive Bayes (TAN) and k-dependence Bayesian classifier (KDB). Although not significant difference is found when using disjoint discretization techniques, i.e. where the intervals do not overlap. It is indeed, as far as AODE and HAODE is concerned, when a non-disjoint technique is applied, as shown in Chapter 7.

- Part IV (Chapter 8) shows the study oriented to find the domains of competence of different semi-naive BNCs both for continuous and discrete domains.

- Part V (Chapter 9) contains the main conclusions of this dissertation and future work is delineated. It also contains the list of publications this thesis has contributed to the existing literature.

Chapter 2

Preliminaries and notation

To be uncertain is to be uncomfortable, but to be certain is to be ridiculous.
Chinese Proverb.

2.1 Bayesian network classifiers

Supervised classification is one of the most popular and, hence, important tasks in data mining. In this context, the basic classification task involves learning a model (or generalization) from a set of labelled data, in order to assign one label to every new example. The model learning phase can be more or less complex, to such a degree that most of the work might be carried out in the inference phase (as in lazy classifiers), often simply called classification phase. Note that as the world is not deterministic, we will have to manage with uncertainty in classification in most of the cases.

Formally, a model is learnt from a dataset with m examples and n attributes, all of them with known labels for a special attribute called class, C. Hence, it is also often referred to as supervised classification in contrast to unsupervised classification or clustering, where the labels are not known a priori. For every example of the type $\vec{e} = \{a_1, a_2, \ldots, a_n\}$, where each a_i is the value for the attribute A_i, a typical classifier would assign a label c_i from a finite set, Ω_c, of possible labels. Note that we are referring here to a finite set of labels, if the class to predict is of a continuous type it becomes a problem of regression (numerical prediction), which is out of the scope of this thesis.

As indicated in Chapter 1, there exist multiple paradigms for classification. In this study we focus on BNs, which might be seen as a combination of statistical techniques and graphical models. Quoted from Russell & Norvig [2009]:

> The Bayesian network formalism was invented to allow efficient representation of, and rigorous reasoning with, uncertain knowledge. This approach largely overcomes many problems of the probabilistic reasoning systems of the 1960s and 1970s; it now dominates artificial intelligence research on uncertain reasoning and expert systems. The approach allows for learning from experience, and it combines the best of classical artificial intelligence and neural nets.

Bayesian technology has become popular and well-established, as demonstrated by the numerous companies specialising in this formalism. Just to give a few examples: Hugin[1], Agenarisk[2], BayesiaLab[3] or Bayesian Intelligence[4]. The commercial interest on BNs suggests they are useful in practical applications. Some domains where BN classifiers have been successfully applied are the following:

Computing and Robotics: It is quite logical and natural that BNCs, developed by computing researchers, were firstly applied to solve certain tasks related to computers such as: e-mail services [Sahami *et al.*, 1998], web/text classification [Jiang *et al.*, 2005] and artificial vision [Rehg *et al.*, 1999]. We can even find a chess player, BayesChess [Fernández & Salmerón, 2008a]. BNCs have also been successfully applied to fault detection in networking systems, with similar aims as those applied in medicine or bioinformatics [Armañanzas, 2009].

[1]`http://www.hugin.com/`
[2]`http://www.agenarisk.com/`
[3]`http://www.bayesia.com/`
[4]`http://www.bayesian-intelligence.com/`

BNCs have been so broadly used in spam filters that many of the commercial-use programs are based in this technology. We can therefore speak about an outstanding family called Bayesian spam filtering. These filters can be integrated into the mail client or separately installed in a filtering software package, for instance SpamAssassin[1] and SpamBayes[2].

Medicine and health care: BNCs have proved to be very useful in many areas of medicine, including heart diseases [Qazi *et al.*, 2007], cancer diagnosis [Antal *et al.*, 2003], gene identification [Armañanzas *et al.*, 2008] and human biology [Morales *et al.*, 2008].

Economy, finance and banking : BNCs have been applied to assess risk in financial operations such as credit approval or deciding whether to invest in a particular area or enterprise. Pavlenko & Chernyak [2010] worked with data from a private mid-sized bank in Ukraine with the aim to design a credit risk model in which a particular role of the related borrowers exposure can be analysed as a risk-aggregating factor. TAN and KDB classifiers are initially constructed with the collaboration of experts. Afterwards, they are also validated and updated according to the results of the assessment process. In Korb & Nicholson [2010, Chapter 7] other examples of credit risk assessment with BNs are reviewed.

Environmental Science: It is another area where BNCs have been successfully applied in the past, and its interest in these Bayesian structures has increased enormously during the last five years. Reviewing the recent literature we can find relevant works in ecology, microbiology [Wang *et al.*, 2007], fish recruitment [Fernandes *et al.*, 2010], fish classification [Axelson *et al.*, 2009], meteorology [Hruschka-Jr. *et al.*, 2005], habitat characterization and conservation planning [Aguilera *et al.*, 2010], Geographical Information Systems and mapping [Porwal *et al.*, 2006] or agriculture [Bressan *et al.*, 2009], among many others.

[1]`http://spamassassin.apache.org/`
[2]`http://spambayes.sourceforge.net/`

BNs provide several advantages to the classification task:

- The networks store information about existing dependencies among the variables involved, which provide a graphical representation capable of inherently dealing with uncertainty, encoded through the probability theory underneath.

- The graphical representation through the BN facilitates the interpretation and formulation of conclusions about the domain of study.

- In addition, BNCs can combine causal relationships with probabilistic logic, which helps to incorporate expert knowledge into the model.

Thus, one of the greatest advantages of the BNs is that they can represent both the qualitative and the quantitative aspect of the problem. The former is encoded in a directed acyclic graph (DAG), whereas the latter involves storing a probability distribution for every node conditioned on its parents. Even though the conditional probability distribution can be represented in several ways, the most common representation is the use of tables, i.e. conditional probability tables (CPTs).

In a DAG, each node represents a variable; an arc represents a direct dependence between the pair of nodes connected. If there is a directed arc from X to Y, it means that X is the parent of Y and Y is child of X. Furthermore, if there exists a directed path from X to Z, it implies that X is an ancestor of Z, while Z is a descendant of X.

Through the local Markov property, which states that a node is conditionally independent of its non-descendant given its parents, we can represent the joint probability distribution of a BN by the product of the CPTs associated with each of its nodes.

In classification, we want to obtain $p(c|\vec{e})$ $\forall c \in \Omega_c$, i.e. the conditional probability for C given $\vec{e}$. The accurate estimation of the probabilities a posteriori for every combination of the class labels and the values of the attributes is infeasible in practice, as it requires a large amount of training data even with a moderate number of attributes. That is why it is convenient to resort to the Bayes theorem:

$$p(c|\vec{e}) = \frac{p(\vec{e}|c)p(c)}{p(\vec{e})}. \tag{2.1}$$

When the probabilities a posteriori are compared for the different class labels, the denominator is constant and can be ignored. The search for the label c^* that maximizes these probabilities is called maximum a posteriori (MAP) rule:

$$c^* = argmax_{c \in \Omega_C}\ p(c|\vec{e}) = argmax_{c \in \Omega_C}\ p(c)p(\vec{e}|c). \tag{2.2}$$

The prior probability $p(c)$ can be easily estimated from training data by calculating the fraction of examples that belong to each class. Different approximations can be used in order to estimate the probabilities conditioned to the class, $p(\vec{e}|c)$, and depend on the structure of the DAG learnt.

The MAP rule, in this case, is equivalent to the 0/1 loss function, defined as: $argmin_{c \in \Omega_C}\ \sum_{c' \in \Omega} L(c, c')p(c'|\vec{e})$, where $L(c, c') = 0$, if $c = c'$; and 1, otherwise.

In this thesis we don't cover general BN structure learning. Learning the structure of a network can take a long time and effort, especially for datasets of high dimensionality. That is why it is often convenient to consider a partially or totally pre-fixed structure from which the CPTs are learnt. The most simple of these structures is the one used by NB, that assumes all the attributes are independent given the class. In spite of its naive assumption, it performs surprisingly well in some domains. As indicated above, many techniques improve the accuracy of NB by alleviating the attribute interdependence problem. All these techniques, known as semi-naive BNCs, do not perform structural search or this search is very simple.

The natural domain of BNs are the discrete variables, hence, we will assume this property for all the variables in the dataset except in Chapters 4 to 7 of this dissertation. It is also often the presence of missing values in data, and even if there exist several methods to replace these values (imputation of single or multiple values, via mean substitution or linear interpolation for example), the direct way to proceed in a BNC is ignoring the affected counting in every CPT where the attribute whose value is missing appears, which is called available-case analysis. Note that this is not the same as ignoring the whole instance (aka complete-case analysis). There are several other options, and it is not clear which is the best way to proceed; it partially depends on the number of samples and proportion of cases with missing data. For more information on how to deal with missing values please refer to Zhang & Lu [2002].

2.1.1 Naive Bayes

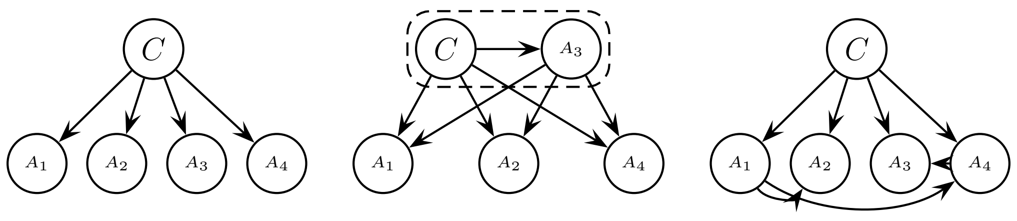
NB [Duda & Hart, 1973] estimates the class conditional probability assuming that all attributes are conditionally independent given the value of the class, and this implies the following factorization: $\forall c \in \Omega_C \; p(\vec{e}|c) = \prod_{i=1}^{n} p(a_i|c)$ (see Figure 2.1 (a)). This approach is more feasible, as a large training set is not required to obtain an acceptable probability estimation. Here, the MAP hypothesis is used to classify as follows:

$$c^* = argmax_{c \in \Omega_C} \; p(c|\vec{e}) = argmax_{c \in \Omega_C} \left(p(c) \prod_{i=1}^{n} p(a_i|c) \right). \tag{2.3}$$

At *training time*, NB has a time complexity $\mathcal{O}(mn)$, where m is the number of training examples. The space complexity is $\mathcal{O}(cnv)$ where v is the average number of values per attribute and c the number of classes. The resulting time complexity at *classification time* is $\mathcal{O}(cn)$, while the space complexity is $\mathcal{O}(cnv)$.

2.1.2 Averaged One-Dependence Estimators

AODE [Webb *et al.*, 2005] is considered an improvement on NB and an interesting alternative to other attempts such as Lazy Bayesian Rules (LBR) [Zheng & Webb, 2000] and Super-Parent TAN (SP-TAN) [Keogh & Pazzani, 1999], since they offer similar accuracy values, but AODE is significantly more



(a) Naive Bayes (b) SPODE (c) TAN/KDB1

Figure 2.1: Examples of network structures with 4 predictive attributes for the following BN classifiers: NB, SPODE (e.g. AODE) and TAN or KDB1 ($k = 1$).

efficient at classification time compared with the first one and at training time compared with the second. In order to maintain efficiency, AODE is restricted to exclusively use 1-dependence estimators. Specifically, AODE can be considered as an ensemble of SPODEs (Superparent One-Dependence Estimators), because every attribute depends on the class and another shared attribute, designated as superparent.

Graphically, every SPODE model used in AODE has a structure such as the one depicted in Figure 2.1 (b), where AODE combines all possible classifiers with this pattern structure. Hence, AODE computes the average of the n possible SPODE classifiers (one for each attribute in the database):

$$c^* = argmax_{c\in\Omega_C}\left(\sum_{j=1,N(a_j)>q}^{n} p(c,a_j)\prod_{i=1,i\neq j}^{n} p(a_i|c,a_j)\right), \qquad (2.4)$$

where the condition $N(a_j) > q$ is used as a threshold to avoid making predictions from attributes with few observations[1].

At *training time*, AODE has a $\mathcal{O}(mn^2)$ time complexity, whereas the space complexity is $\mathcal{O}(c(nv)^2)$. The resulting time complexity at *classification time* is $\mathcal{O}(cn^2)$, while the space complexity is $\mathcal{O}(c(nv)^2)$.

AODE offers an attractive alternative to other approaches that aim to improve NB maintaining its efficiency, as it provides competitive error rates with an efficient profile [Zheng & Webb, 2005].

[1]In all of our experiments in the following chapters, this q value has been set to 1, which is the default value in WEKA [Hall *et al.*, 2009; Witten & Frank, 2005].

In Cerquides & de Mántaras [2005], two algorithms based on expectation maximization and on constrained optimization for learning maximum a posteriori weights for the different SPODEs are proposed. These two proposals improve over uniform aggregation and Bayesian model averaging [Hoeting *et al.*, 1999, 2000]. Another simple and efficient way of weighting AODE is known as weightily averaged one-dependence estimators (WAODE) [Jiang & Zhang, 2006], where the different models are weighted according to the mutual information between the superparent and the class.

2.1.3 Other semi-naive Bayesian classifiers

2.1.3.1 Tree augmented naive Bayes (TAN)

The TAN model [Friedman *et al.*, 1997] relaxes the conditional independence restriction without a large increase in the complexity of the construction process. The idea behind TAN entails learning a maximum weighted spanning tree [Chow & Liu, 1968] based on the conditional mutual information between two attributes given the class label, choosing a variable as root and completing the model by adding a link from the class to each attribute. This procedure is described in more detail in Algorithm 2.1.

The mutual information between two discrete variables: A_i and A_j conditioned on the class C can be defined as:

$$MI(A_i, A_j; C) = \sum_{i=1} \sum_{j=1} \sum_{r=1} p(a_i, a_j, c_r) \frac{\log p(a_i, a_j, c_r)}{p(a_i|c_r)p(a_j|c_r)}. \tag{2.5}$$

Friedman *et al.* [1997] guarantee that the tree learned from the training data is the optimal one, i.e., it is the best possible probabilistic representation from the available data as a tree. TAN becomes a structural augmentation of NB where every attribute has the class variable and at most one other attribute as its parents, see Figure 2.1 (c). It is considered a fair trade-off between model complexity and model accuracy.

At *training time*, TAN generates a three-dimensional table, with a space complexity $\mathcal{O}(c(nv)^2)$. The time complexity of forming the three-dimensional probability table is $\mathcal{O}(mn^2)$ and $\mathcal{O}(cn^2v^2)$ of creating the parent function. A

Algorithm 2.1: The TAN algorithm

```
   Input: Dataset with variables $A_1, \ldots, A_n, C$.
   Output: TAN model.
 1 Construct a complete undirected graph (UG), $\mathcal{U}$, with nodes
   $A_1, \ldots, A_n$.
 2 Label each arc $(A_i, A_j)$ with the conditional mutual information
   between $A_i$ and $A_j$ given $C$, i.e., $MI(A_i, A_j; C)$.
 3 Maximum Weight Spanning Tree Algorithm:
 4 begin
 5     Let $\mathcal{G}$ be an empty UG with nodes $A_1, \ldots, A_n$.
 6     while Number of arcs in $\mathcal{G}$ is $\leq n - 1$ do
 7         Add the arc with maximum weight if not cycle in $\mathcal{G}$.
 8     end
 9 end
10 Transform $\mathcal{G}$ into a directed graph (DG), $\mathcal{T}$, selecting a root.
11 Complete $\mathcal{T}$ by adding $C$ and arcs $(C, A_i)$ $\forall i$.
12 Compute the CPTs.
13 Let TAN be a BN with structure $\mathcal{T}$ and probability distributions in
   CPTs.
14 return TAN
```

maximal spanning tree is then generated, with time complexity $\mathcal{O}(n^2 \log n)$. At *classification time*, TAN only needs to store the probability tables, with space complexity $\mathcal{O}(cnv^2)$. The time complexity of classifying a single example is $\mathcal{O}(cn)$.

In fact, the TAN classifier can be considered a particular case of the forest augmented naive Bayes (FAN) [Lucas, 2004]. This classifier aims to alleviate TAN's limitation that comes from the fact that some arcs may imply the insertion of noise, as the number of arcs in the tree has to be $n - 1$ (where n is the number of attributes). In FAN, a maximum spanning forest is learnt, which implies to form a forest of trees with a disjoint set of attributes. The number of arcs to include must be previously specified so that it is possible to ignore certain arcs, hence becoming a more flexible model than TAN.

2.1.3.2 k-dependence Bayesian classifier (KDB)

Sahami [1996] introduced the notion of k-dependence estimators, from which the probability of each attribute value is conditioned by the class and, at most, k other attributes. Throughout the KDB algorithm (shown below) it is possible to construct classifiers across the whole spectrum, from the NB structure to the full BN structure, by varying the value of k, i.e. the maximum number of parents that every attribute can have.

Algorithm 2.2: The KDB algorithm

```
   Input: Dataset with variables $A_1, \dots, A_n, C$ and $k$.
   Output: KDB model.
1  Calculate $MI(A_i; C)$ for all attributes.
2  Calculate $MI(A_i, A_j; C)$ for each pair of attributes $(i \neq j)$.
3  Let the used variable list be $U = \emptyset$.
4  Let $\mathcal{G}$ be a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which $\mathcal{V}$ is a set of vertices and $\mathcal{E}$ is a
   set of links.
5  $\mathcal{V} = \{C\}$
6  $\mathcal{E} = \emptyset$
7  while $(\exists A_i \notin U)$ do
8      $A_{max} = max_i\{MI(A_i; C)\}, \forall A_i \notin U$
9      $\mathcal{V} = \mathcal{V} \cup A_{max}$
10     $\mathcal{E} = \mathcal{E} \cup (C, A_{max})$
11     $vk = min(|U|, k)$
12     Let the auxiliary variable list be $Q = \emptyset$
13     while $(vk > 0)$ do
14         $A_{max2} = max_j\{MI(A_{max}; A_j|c)\}, \forall A_j \in U \wedge A_j \notin Q$
15         $\mathcal{E} = \mathcal{E} \cup (A_{max2}, A_{max})$
16         $Q = Q \cup A_{max2}$
17         $vk = vk - 1$
18     end
19     $U = U \cup A_{max}$
20 end
21 Compute the CPTs.
22 Let KDB be a BN with structure $\mathcal{G}$ and probability distributions in
   CPTs.
23 return $KDB$
```

The advantage of this type of classifiers compared with TAN is their flexibility. In TAN, a variable can have at most one other variable for its parent. This restriction on the number of parents strongly constrains the dependencies that can be modelled between the group of features.

Computing the actual network structure with the KDB algorithm requires $\mathcal{O}(n^2mcv^2)$ and calculating the CPTs within the network takes $\mathcal{O}(n(m+v^2))$, where v here is the maximum number of values that an attribute may take. Classification time would require $\mathcal{O}(nck)$.

Apart from the classifiers mentioned above, there exist other not-so-well-known approaches that should also be taken into account. We indicate some of them in the following subsections.

2.1.3.3 Hidden naive Bayes (HNB)

[Zhang *et al.*, 2005] This classifier creates a hidden parent for each attribute that combines the influences from all other attributes by considering the following classification rule:

$$c^* = argmax_{c\in\Omega_C}\left(p(c)\prod_{i=1}^{n}\prod_{i=1,i\neq j}^{n}W_{ij}p(a_i|a_j,c)\right), \tag{2.6}$$

where $W_{ij} = \frac{MI(Ai,Aj;C)}{\sum_{j=1,j\neq i}^{n} MI(Ai,Aj;C)}$. The HNB is similar in idea to AODE, but it has higher training time $\mathcal{O}(mn^2 + kv^2n^2)$.

2.1.3.4 Full Bayesian network classifier (FBNC)

[Su & Zhang, 2006] This Bayesian classifier assumes a full BN and learns a decision tree for each CPT, with a novel and more efficient algorithm. The authors claim that it is quadratic in training time $O(tn^2)$, and linear in classification time $O(n)$, providing competitive results with other state-of-the-art learning algorithms.

2.1.3.5 Bayesian network augmented naive-Bayes (BAN)

[Friedman *et al.*, 1997] The BAN classifier further relaxes the independence assumption as it creates a BN among the attributes, while it maintains the class variable as a parent of each attribute. The posterior probability of this classifier is formulated as:

$$c^* = argmax_{c\in\Omega_C}\left(p(c)\prod_{i=1}^{n}p(a_i|pa(a_i),c)\right), \tag{2.7}$$

where $p(a_i)$ are the parents of every A_i. It is the empty set for NB, it is a set with one single parent for TAN, and it is an unlimited parent set for BAN.

2.1.3.6 Multinet classifier based on dependency networks (MultiDN)

In Gámez *et al.* [2008], the authors propose the use of dependency networks for classification with methods for reusing calculations across mixture components. This classifier obtains a competitive trade-off between accuracy and learning time when dealing with data sparse classes.

2.2 Alternatives for continuous variables

So far, we have simply considered that all the attributes in the dataset of interest are of discrete (nominal) type. However, in many real applications the input data are of continuous nature. At the moment of writing, there is not a clear guideline on the best way to handle these numeric attributes when learning a Bayesian model.

In general, Bayesian methods make use of multinomial distributions, which assume all the variables are discrete. Hence, the direct way to proceed to be able to treat these numeric attributes is discretization. Even though it entails an unavoidable loss of information, it can be a good (or even the best) alternative in many domains. Other techniques to directly deal with the original numeric values implies assuming that these attributes follow a known parametric distribution, such as Gaussian, kernel or mixture of truncated exponentials.

There exist other alternatives, and it is not always clear which the best option is. The first question raised is whether discretization is suitable for our purposes or we should directly assume our samples follow a known parametric distribution. But even if it was clear that discretization is the best option, the type of discretization along with its configuration values should be set. On the other hand, if no discretization is performed, one or more probability distributions must be selected. The best choice is not always clear, that is why in the following sections we pretend to provide an overview of the most common procedures in Bayesian classifiers: discretization techniques, kernel and Gaussian distributions. In addition, MTEs are included, as it is gaining popularity as a flexible modelling framework for hybrid domains.

2.2.1 Discretization techniques

Every discretization process involves the transformation of continuous domains into discrete counterparts. It implies an unavoidable loss of information, since from the infinite number of continuous values provided as original input, only a finite set of values is kept. In this context, we consider discretization as a data pre-processing technique that transforms a quantitative attribute into a qualitative one. In practice, the discretization process can be viewed as a method for reducing data dimensionality, as input data are transformed from a huge range of continuous values into a much smaller subset of discrete ones. Although we can find a considerable variation in the terminology used to refer to this these types of attributes [Yang, 2003], in this chapter we will refer to attributes for which no arithmetic operations can be applied as *discrete*, and the rest as *continuous* or *numeric* indistinctly.

The necessity of applying discretization on the input data can be due to different reasons. Firstly, many powerful classification and modelling algorithms only operate on categorical or nominal data, and therefore discretization is a prerequisite if we wish to apply these algorithms (e.g. certain Bayesian network methods). In other cases, discretization improves the run time for the given algorithms, such as decision trees, as the number of possible partitions to be evaluated is drastically decreased. Also, discrete values for a variable may on many occasions provide a higher interpretability of the models. Finally, there are cases where discretization is simply a pre-processing step with the aim of obtaining a reduction in the value set and, thus, a reduction in the noise, generally present in the data.

Many different taxonomies for dividing and organizing the various discretization techniques can be found in the literature [Liu *et al.*, 2002]. The most commonly used being the one which distinguishes between **unsupervised** (e.g. equal frequency or width, k-means) and **supervised** (such as those based on entropy [Fayyad & Irani, 1993], Hellinger divergence-based [Lee, 2007], 1R algorithm) methods. This distinction is made depending on whether or not the method takes class information into account in order to find proper intervals. Traditionally, supervised discretization techniques have been believed to be especially suitable for classification tasks, although the results included in this thesis, in relation to this, do not entirely agree.

Another way of categorizing discretization methods is by considering whether variables are discretized independently, known as **univariate** (and also as local); or if not, we encounter the **multivariate** methods (also referred to as global, although we prefer the former nomenclature to avoid ambiguity), which take into consideration the relationships among attributes during discretization. It has been proven that multivariate techniques can produce better discretizations, since the joint information measures are much more powerful [Chmielewski & Jerzy, 1996].

Also, **optimization** methods, such as evolutionary computation techniques, can be used in this multivariate scheme [Flores *et al.*, 2007]. However, these methods cannot be considered under time restrictions, and their complexity increases dramatically with the number of attributes. Thus, the main drawback is that they are much more costly in resource consumption than classical approaches. This fact makes them less attractive for the semi-naive family of BNCs considered in this chapter.

Furthermore, for a particular classifier (or family of classifiers) it is possible to construct **ad hoc** discretization methods, which could be categorized as tailored methods. In this category, we could place two discretization techniques proposed in Yang & Webb [2009] and the more recent hybrid approach by Wong [2012], in principle designed to fit NB's needs.

Nonetheless, all the discretization techniques taken into account so far form non-overlapping intervals for numeric attributes. A novel type of discretization is proposed for NB in Yang & Webb [2002], called non-disjoint discretization (NDD), which creates bins that overlap, providing a good performance for NB. In this thesis we focus on this latter distinction, as we believe it is critical to differentiate between these two to show significant improvement in some of the most well-known semi-naive Bayesian classifiers. Following this division, we are including next, descriptions of some of the most popular disjoint discretization methods, and detailed notation on NDD compared with the traditional disjoint discretization techniques.

2.2.1.1 Most popular disjoint discretization methods

Equal-width discretization [Dougherty *et al.*, 1995]

This is a technique for unsupervised discretization, since the class value is not considered when the interval limits are selected, and is probably the simplest method for discretizing data. Equal-width divides the range of the attribute into b bins of the same width, where b is a parameter supplied by the user. If an attribute A is observed to have values bounded by a_{min} and a_{max}, then this method computes the bin width by:

$$\delta = \frac{a_{max} - a_{min}}{b}. \tag{2.8}$$

Hence, the bin boundaries are set at $a_{min} + i\delta$, where $i = 1, \ldots, b-1$.

It is quite usual to set this value to 5 or 10 bins, although the optimum value for b depends on, among other factors, the dataset size.

The software tool WEKA [Hall *et al.*, 2009; Witten & Frank, 2005] provides the utility of searching for the most appropriate value of b by means of a filter method which minimizes the partition entropy.

Its time complexity is $\mathcal{O}(m)$, m being the number of instances.

Equal-depth (or frequency) discretization [Dougherty *et al.*, 1995]

In this unsupervised technique the values are ordered and divided into b bins so that each one contains approximately the same number of training instances.

Therefore, every bin contains m/b instances with adjacent values. This type of discretization method provides a more balanced discretization in the different bins and usually a more intuitive solution.

As a group of values with identical values must be placed in the same bin, it is not always possible to generate b intervals with exactly the same number of values.

Time complexity for this technique is $\mathcal{O}(m \log m)$, as it is necessary to perform an ordering of the data.

Minimum-entropy-based discretization by Fayyad & Irani [1993] This refers to a supervised technique which evaluates, for cut-off point candidates, those points between every pair of values (usually mean points) which are contiguous in the ordered data and whose class labels change. When evaluating every candidate, data are divided into two intervals and the entropy for each class is computed. A binary discretization is performed at that candidate cut-off point which minimizes the entropy. This process is repeated in a recursive way by applying the Minimum Description Length (MDL) criterion to decide when to stop. Following similar notation to Fayyad & Irani [1993], the specific algorithm is detailed in Algorithm 2.3. Its time complexity is $\mathcal{O}(cm \log m)$.

This method and the two others described above are applied to each continuous attribute independently, and hence are considered univariate discretization techniques.

2.2.1.2 Disjoint vs non-disjoint discretization

Formally, given the numeric attribute values $x_i, x_j \in \mathbb{R}$, any disjoint discretization method would create a unique interval $(a, b] \ni x_i$ and $(d, e] \ni x_j$ for every value so that AODE's statistics, $p(X_j = x_j, C = c)$ and $p(X_i = x_i | C = c, X_j = x_j)$ would be estimated by

$$p(X_j = x_j, C = c) \approx p(d < X_j \leq e, C = c), \tag{2.9}$$

Algorithm 2.3: Fayyad and Irani's discretization method

```
   Input: Set of instances $S$ for feature $A$.
   Output: The cut points in which to discretize $A$.
1  Sort $S$ by increasing value of $A$.
2  Let $T$ be a candidate cut point.
3  repeat
4      Select all $T$ such that there exist two examples: $e_1, e_2 \in S$ with
       different classes where $A(e_1) < T < A(e_2)$; and there exists no
       other example $e'$ for which $A(e_1) < A(e') < A(e_2)$ (usually the
       mean points), where $A(e)$ is the value of $A$ in the instance $e$.
5      Let $S_1 \subset S$ where $A$-values $\leq T$ and $S_2 = S - S_1$.
6      $\forall T$ compute $E(A,T;S) = \frac{S_1}{S} Ent(S_1) + \frac{S_2}{S} Ent(S_2)$, where
       $Ent(S_i) = -\sum_{c \in \Omega}(p_c \log_2(p_c))$.
7      Select $T_{min}$ as $min_i E(A, T_i; S)$.
8      $Threshold = \frac{\log_2(t-1)}{t} + \frac{\Delta(A,T_{min};S)}{t}$.
9      $Gain(A, T_{min}; S_i) = Ent(S_i) - E(A, T_{min}; S_i)$.
10     Recursively perform binary discretization in $\mathbf{S = S_j}$.
11 until
   $Gain(A, T_{min}; S_1) \leq Threshold$ or $Gain(A, T_{min}; S_2) \leq Threshold$ ;
12 return All the $T_{min}$ obtained as cut points.
```

$$p(X_i = x_i | C = c, X_j = x_j) \approx p(a < X_i \leq b | C = c, d < X_j \leq e). \qquad (2.10)$$

In disjoint discretization techniques (e.g. equal frequency or equal width division, MDL, etc.) every numeric sample belongs to a single interval. I.e., considering $x_i < x_j$, if $a \neq d$ (they do not fall in the same interval) then $d \geq b$. This implies that for those cases where the original numeric value falls around the centre of the interval assigned, we could expect more distinguishing information than when it falls near one of the boundaries of the interval. In the latter situation, it is more questionable to substitute $p(X_j = x_j, C = c)$ by $p(d < X_j \leq e, C = c)$ for example.

In contrast, NDD creates bins that overlap. Then, numeric values are always located toward the middle of the interval to which they belong. The idea justifies since the test instances are independent of each other, and so it is not required to form a uniform set of disjoint intervals for a numeric attribute. Instead, it should form an appropriate interval to the single value offered by a particular test instance.

Lazy discretization [Hsu *et al.*, 2000] also places a value in the middle of an interval. However, it has a low computational efficiency, given its lazy methodology. NDD, in contrast, is more efficient as it creates the intervals at training time.

Non-Disjoint Discretization [Yang & Webb, 2002]

NDD is an unsupervised technique that forms t atomic intervals $B_1 = [a'_1, b'_1]$, $B_2 = (a'_2, b'_2], \ldots, B_t = (a'_t, b'_t]$ (where $b'_i = a'_{i+1}, \forall i$), with equal frequency. In its definition for NB [Yang & Webb, 2002], one operational interval or label is formed then for each set of three consecutive atomic intervals, such that the rth $(1 \leq r \leq t-2)$ interval $(a_r, b_r]$ satisfies $a_r = a'_r$ and $b_r = b'_{r+2}$. Each numeric value x is assigned to interval $(a'_{i-1}, b'_{i+1}]$ where i is the index of the atomic interval $(a'_i, b'_i]$ such that $a'_i < x \leq b'_i$, except when $i = 1$ in which it is assigned to interval $[a'_1, b'_3]$ and when $i = t$ that it is assigned to interval $(a'_{t-2}, b'_t]$. Here t and the number of instances per atomic interval s are selected proportionally to the number of training instances, following the idea of proportional k-interval discretization [Yang & Webb, 2001]. That is, $t = s \approx \lfloor\sqrt{m}\rfloor$, each operational interval then having $3s$ samples. Figure 2.2 shows a graphical example of the partition.

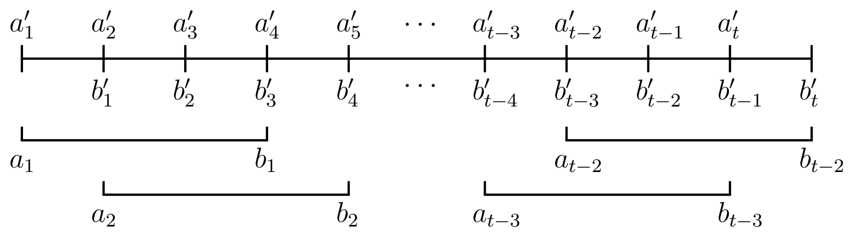


Figure 2.2: Example of NDD discretization for NB.

NDD is dominated by sorting as well, and hence, its complexity is also $\mathcal{O}(m \log m)$.

2.2.2 Conditional Gaussian networks

Continuous variables in a BN can be modelled by a Gaussian distribution function (also called normal distribution). Any Gaussian distribution may be defined by two parameters, location and scale: the mean ("average", μ) and variance (standard deviation squared, σ^2) respectively. Likewise, every continuous node can have a Gaussian distribution for every configuration of its discrete parents. If a continuous node has one or more continuous nodes as parents, the mean can be linearly dependent over the states of these continuous parents. This is the basic idea underlying CGNs [Lauritzen & Jensen, 2001]. Note that discrete nodes are not allowed to have continuous parents though.

In this case, a parametrical learning process is carried out, where the estimation of the parameters is made from data. These parameters are modelled by the dependency relationships between variables, represented by the structure of the corresponding classifier or BN. A noteworthy property of CGNs is that they offer a frame where exactitude in inference is guaranteed. Another advantage of Gaussian networks is that they only need $\mathcal{O}(n^2)$ parameters to model a complete graph.

In general, every node stores a local density function (linear regression model) where the distribution for a continuous variable X with discrete parents $\boldsymbol{Y}$ and continuous parents $\mathbf{Z} = \{Z_1, \ldots, Z_s\}$ (with s the number of continuous parents) is a one-dimensional Gaussian distribution over the states of its parents [DeGroot, 1970]:

$$f(X|\mathbf{Y} = y, \mathbf{Z} = z; \Theta) = \mathcal{N}(x : \mu_X(y) + \sum_{j=1}^{s} b_{XZ_j}(y)(z_j - \mu_{Z_j}(y)), \sigma^2_{X|\mathbf{Z}}(y)), \tag{2.11}$$

where:

- $\mu_X(y)$ is the mean of X with the configuration $Y = y$ of its discrete parents.
- $\mu_{Z_j}(y)$ is the mean of Z_j with the configuration $Y = y$ of its discrete parents.

- $\sigma^2_{X|\mathbf{Z}}(y)$ is the conditional variance of X over its continuous parents $\mathbf{Z}$ and also according to the configuration $Y = y$ of its discrete parents.
- $b_{XZ_j}(y)$ is a regression term that individually measures the strength of the connection between X and every continuous parent (it will be equal to 0 if there is not an edge between them).

The local parameters are given by $\Theta = (\mu_X(y), b_X(y), \sigma^2_{X|\mathbf{Z}}(y))$, where $b_X(y) = (b_{XZ_1}(y), \ldots, b_{XZ_{s(y)}})^t$ is a column vector.

Then, if we focus on the bivariate case, where the X variable is only conditioned by one continuous variable Z and the discrete variables mentioned, the conditional variance and the regression term would be easily obtained as shown in Equations 2.12 and 2.13[1]:

$$\sigma^2_{X|Z}(y) = \sigma^2_X(y) - b^2_{XZ}(y)\sigma^2_Z(y), \tag{2.12}$$

$$b_{XZ}(y) = \frac{\sigma_{XZ}(y)}{\sigma^2_Z(y)}. \tag{2.13}$$

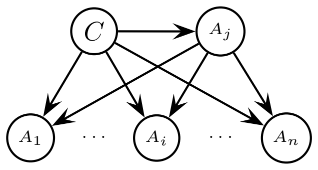
Figure 2.3 shows an example of factorization of the density function in a SPODE structure, as in the model depicted on the left. Following the former notation, in this case $b_X(y) = b_{XZ}(y)$, as there is just one continuous variable.

Equation 2.11 has been obtained following the guidelines in Larrañaga *et al.* [1999] and Neapolitan [2003]. However, in the Hugin tool [Andersen *et al.*, 1989], the estimation of the Gaussian distribution of interest, is carried out in a slightly different way. In the estimation of the final mean for the CGN, Hugin does not take into account the means of the continuous parents, and the variance is constant for every configuration state of the discrete parents. Hence, the corresponding equation according to Hugin principles

[1]The estimate in Equations 2.12 and 2.13 has been obtained by working out the value of $\sigma^2_{X|Z}(y)$ and $b_{XZ}(y)$ when the inverse of the precision matrix (W^{-1}) and the covariance matrix (Σ) from the Gaussian network are matched:

$$W^{-1} = \begin{pmatrix} \sigma^2_Z(y) & b_{XZ}(y)\sigma^2_Z(y) \\ b_{XZ}(y)\sigma^2_Z(y) & \sigma^2_{X|Z}(y) + b^2_{XZ}(y)\sigma^2_Z(y) \end{pmatrix} = \begin{pmatrix} \sigma^2_Z(y) & \sigma_{XZ}(y) \\ \sigma_{XZ}(y) & \sigma^2_X(y) \end{pmatrix} = \Sigma$$

Structure



Local densities

$$\Theta_c = (CPT) \qquad f_C \rightsquigarrow P(C)$$
$$\Theta_j = (\mu_j(c), -, \sigma_j(c)) \qquad f_{A_j|C=c} \rightsquigarrow \mathcal{N}(\mu_j(c), \sigma_j(c))$$
$$\Theta_1 = (\mu_1(c), b_1(c), \sigma_{1|j}(c)) \qquad f_{A_1|C=c,A_j=a_j} \rightsquigarrow \mathcal{N}(\mu_1(c) + b_{1j}(c)(a_j - \mu_j(c))), \sigma_{1|j}(c))$$
$$\Theta_i = (\mu_i(c), b_i(c), \sigma_{i|j}(c)) \qquad f_{A_i|C=c,A_j=a_j} \rightsquigarrow \mathcal{N}(\mu_i(c) + b_{ij}(c)(a_j - \mu_j(c))), \sigma_{i|j}(c))$$
$$\Theta_n = (\mu_n(c), b_n(c), \sigma_{n|j}(c)) \qquad f_{A_n|C=c,A_j=a_j} \rightsquigarrow \mathcal{N}(\mu_n(c) + b_{nj}(c)(a_j - \mu_j(c))), \sigma_{n|j}(c))$$

Factorization of the joint density function

$$f(c, a_j, a_1, \ldots, a_i, \ldots, a_n) = f(c)f(a_j|c)f(a_1|c, a_j)\cdots f(a_i|c, a_j)\cdots f(a_n|c, a_j) = p(c)\frac{1}{\sqrt{2\pi}\sigma_j(c)}e^{-\frac{1}{2}\left(\frac{a_j-\mu_j(c)}{\sigma_j(c)}\right)^2}$$
$$\frac{1}{\sqrt{2\pi}\sigma_{1|j}(c)}e^{-\frac{1}{2}\left(\frac{a_1-(\mu_1(c)+b_{1j}(a_j-\mu_j(c)))}{\sigma_{1|j}(c)}\right)^2}\cdots\frac{1}{\sqrt{2\pi}\sigma_{i|j}(c)}e^{-\frac{1}{2}\left(\frac{a_i-(\mu_i(c)+b_{ij}(a_j-\mu_j(c)))}{\sigma_{i|j}(c)}\right)^2}\cdots$$
$$\cdots\frac{1}{\sqrt{2\pi}\sigma_{n|j}(c)}e^{-\frac{1}{2}\left(\frac{a_n-(\mu_n(c)+b_{nj}(a_j-\mu_j(c)))}{\sigma_{n|j}(c)}\right)^2}$$

Figure 2.3: Structure, local densities and result from the factorization of the joint density function in a network with the SPODE structure where all the predictive attributes are continuous.

would be as follows:

$$f(X|\mathbf{Y} = y, \mathbf{Z} = z; \Theta) = \mathcal{N}(x : \mu_X(y) + \sum_{j=1}^{s} b_{XZ_j}(y)z_j, \sigma_X^2(y)). \qquad (2.14)$$

On the other hand, in Pérez *et al.* [2006] the authors consider Equation 2.11, but the variance for the CGN is constant for every configuration state of the discrete parents.

It is also in this work [Pérez *et al.*, 2006], that the authors show how to adapt different classifiers, in which we find NB, TAN, KDB and semi naive Bayes, and other proposals based on feature selection from these classifiers, to the conditional Gaussian network paradigm, along with the corresponding empirical evaluation. It is also interesting to note their proposal to calculate the mutual information between every pair of continuous predictive variables conditioned on the class.

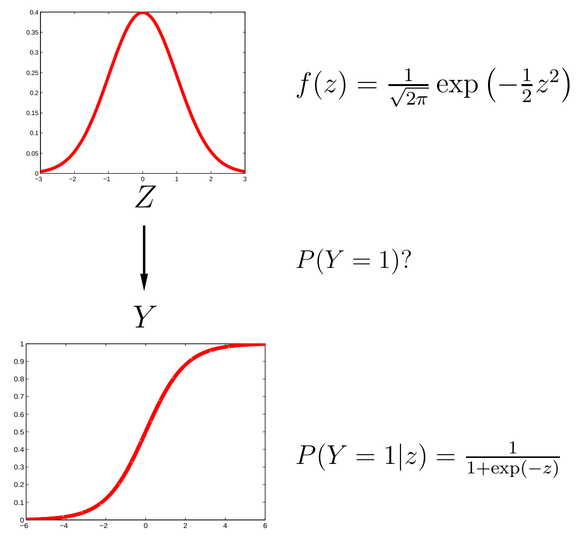

$f(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}z^2\right)$
Z
$P(Y = 1)?$
Y
$P(Y = 1|z) = \frac{1}{1+\exp(-z)}$

Figure 2.4: Example of illegal configuration in CGNs with two variables: a discrete variable Y with a continuous parent Z.

So far, we have just considered how to model continuous variables conditioned on either discrete or continuous variables as well, but how can we face discrete variables with continuous parents? In this work we do not allow discrete variables to have continuous parents. This for example ensures availability of exact local computation methods, see Lauritzen [1992]; Lauritzen & Jensen [2001]. Figure 2.4 shows an example of what we can consider an illegal configuration for CGNs: a discrete variable Y with a continuous parent Z.

Furthermore, CGNs should be preferably used when Gaussian data are provided. In principle, it may seem easy to determine whether the data of interest follows a Gaussian distribution and hence, deciding whether or not to use CGN for our classifiers. There exist several statistical tests to check normality, such as Kolmogrov-Smirnov or ShapiroWilk, available through different tools. Nevertheless, it is important to take into account the structure of the BN we are considering for classification and performing multivariate normality tests according to it.

2.2.3 Kernel density estimation

Modelling all the attributes in a dataset through Gaussian estimations can be inaccurate, if the group of samples for all or some of the attributes does not follow a normal distribution. A possible solution to this problem is the use of **histograms**, as they are considered the most simple non-parametric density estimators. Unlike the parametric estimators, where the estimator has a prefixed function and the parameters of that function are the only information to store, the non-parametric estimators do not have a prefixed structure and depend on all the samples to provide estimation. In order to build a histogram, the range of the data is divided into subintervals of equal size (bins), and they are represented in the X-axis. For every sample belonging to a specific bin, the corresponding block in the Y-axis is incremented by one unit.

Nevertheless, the use of histograms has several problems, such as the lack of smoothing, the dependence of the bin-width and the final points selected. In order to ameliorate these issues, we can resort to the use of **density estimators based on kernels** [Bernard, 1986]. To relieve the dependence on the final points selected for each bin, estimations based on kernels build a kernel function for every sample. It is possible to smooth density estimation by using an smoothed kernel function; hence avoiding two out of the three above-mentioned problems in histograms. The binwidth issue can also be solved, as we will introduce below.

Formally, kernel estimations smooth the contribution of each sample according to the points in its neighbourhood. The contribution of the point $x(i)$ on the estimation of other point x depends on how separate they are. The scope of this contribution also depends on the shape and width adopted by the kernel function K. The estimated density in the point x is defined through the following equation:

$$\hat{f}(x) = \frac{1}{mh} \sum_{i=1}^{m} K\left(\frac{x - x(i)}{h}\right), \tag{2.15}$$

where m is the number of samples and $h > 0$ is a smoothing parameter called the bandwidth. Intuitively, one wants to choose h as small as the data allows, however there is always a trade-off between the bias of the estimator and its variance.

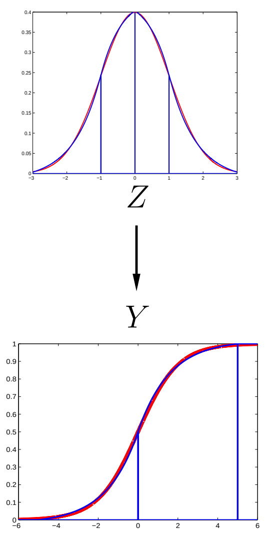
Gaussian kernels are the most well known, but there exist other options, such us uniform kernels, triangulars', Epanechnikov's, etc. Even though the selection of K determines the shape of the density to estimate, the literature suggests that this selection is not critical, at least among the most common ones [Deaton, 1997]. It is believed that the specification of the bandwidth is even more important: the bigger the value of h, the greater the smoothing factor is.

From its definition, one can deduce that both temporal and space complexity in kernel estimations depend also on the number of instances of the dataset. Hence, it imposes an additional undesired restriction to the application of semi-naive Bayesian classifiers considered in this thesis, where keeping time and space constrains under control is one of the main goals.

Still, we can find studies in literature that successfully apply kernel estimations to semi-naive Bayesian classifiers, in the sense that if no constrains in terms of space or time are imposed, the results are generally much better than other alternatives. In John & Langley [1995] the authors introduce the notion of flexible classifier, similar to NB except for the method used for density estimation on continuous variables. They particularly investigate kernel density estimation with Gaussian kernels, selecting $h = \sigma_c = \frac{1}{\sqrt{m_c}}$ as bandwidth, where m_c is the number of training instances with class c. It is in Pérez *et al.* [2009], where the generalization of this notion of flexible naive Bayes is proposed, extended to other paradigms such as TAN, KDB or even the complete graph classifier. It is also interesting to note the new definition for an estimator of the mutual information based on kernels.

2.2.4 Mixture of truncated exponentials

Even though CGNs offer a frame where it is possible to guarantee the exactitude in the inference under time constrains, there exists a serious restriction: it is not possible to model discrete variables with continuous parents. Fur-



$$f(z) = \begin{cases} -0.0172 + 0.931e^{1.27z} & \text{if } -3 \le z < -1 \\ 0.442 - 0.0385e^{-1.64z} & \text{if } -1 \le z < 0 \\ 0.442 - 0.0385e^{1.64z} & \text{if } 0 \le z < 1 \\ -0.0172 + 0.9314e^{-1.27z} & \text{if } 1 \le z < 3 \end{cases}$$

Calculate $P(Y = 1)$ with MTEs: $P(Y = 1) \approx 0.4996851$

$$P(Y = 1|z) = \begin{cases} 0 & \text{si } z < -5 \\ -0.0217 + 0.522e^{0.635z} & \text{if } -5 \le z < 0 \\ 1.0217 - 0.522e^{-0.635z} & \text{if } 0 \le z \le 5 \\ 1 & \text{si } z > 5 \end{cases}$$

Figure 2.5: Example of the use of MTEs to model a discrete variable Y with a continuous parent Z.

thermore, this model is especially useful in those situations where the joint distribution of the continuous variables given the configuration of its discrete parents follows a multivariate Gaussian; nevertheless, it is possible to find scenarios where this hypothesis is not accomplished. In order to overcome this problem there is a relatively new alternative that is becoming more and more popular, the use of Mixtures of Truncated Exponentials (MTEs) [Moral *et al.*, 2001]. MTEs can be an attractive alternative to discretization, as discretization can be seen as an approximation to a density function with a mixture of uniforms, being the use of exponentials a more accurate estimation. Figure 2.5 shows an example of the use of MTEs to model the illegal configuration presented in Figure 2.4.

Following the former notation, where $Y = Y_1, \dots, Y_d$ is the set of discrete variables and $Z = Z_1, \dots, Z_c$ the set of continuous variables, and T both, with $d + c = n$. Considering that for the classification task $Y \neq \emptyset$, as at least the class variable is discrete, a function $f : \Omega_T \mapsto \Re_0^+$ is an MTE potential if for each value $y \in \Omega_Y$, the potential over the continuous variables Z is defined as follows:

- For every value $y \in \Omega_Y$, the density function $f_y(z) = f(y, z)$ is defined as follows:

$$f_y(z) = a_0 + \sum_{i=1}^{l} a_i \; exp\left\{\sum_{j=1}^{c} b_i^{(j)} z_j\right\}, \tag{2.16}$$

where all $z \in \Omega_Z$, $a_i \in \mathbb{R}$ and $b_i \in \mathbb{R}^c$, $i = \{1, \ldots, l\}$.

We also say that f is an MTE potential if there is a partition $D_1, \ldots, D_k$ of Ω_Z into hypercubes and in each partition, f is defined as in Equation 2.16. An MTE potential is an MTE density if it integrates to 1.

In a BN, two types of densities can be found:

- $f(x)$ for each variable X with no parents.
- A conditional density $f(x|pa(x))$ for each variable X with parents $pa(X)$.

A conditional density $f(x|pa(x))$ is an MTE potential that obtains a density function for X when the possible values for $pa(X)$ are fixed. Note that either X or its parents can be discrete or continuous.

2.2.4.1 Estimations of univariate and conditional MTEs:

If we restrict the definition of an MTE potential to a variable with no parents and it is restricted to a single constant term and two exponentials, we obtain the following densities:

$$f^*(x) = k + a \; exp\{bx\} + c \; exp\{dx\}. \tag{2.17}$$

The estimation of the parameters $(\hat{a}, \hat{b}, \hat{c}, \hat{d}, \hat{k})$ of a univariate MTE density function is carried out through the MTE-fitting algorithm, described in Rumí *et al.* [2006]. However, this method is not valid for the conditional case, as more restrictions should be considered over the parameters in order to force the integration of the MTE potential for each combination of the values for $pa(X)$. Precisely, the use of conditional distributions through MTEs applied to the learning phase in a NB classifier is shown in Rumí *et al.* [2006]. In this case, the adaptation of the MTE-fitting algorithm is straightforward, as it is called for every class value and the marginal function of the class is estimated according to its frequency. Nevertheless, this method is only valid if the variable whose distribution we want to estimate has discrete parents. Hence, it is not extensible to appearance of numeric parents.

This problem was in fact already solved in Moral *et al.* [2003], where the authors propose to partition at the domain of the conditioning variables and adjust the univariate density function for each part using the MTE-fitting algorithm. More precisely, the algorithm learns a mixed tree whose leaves contain MTE densities that only depend on the child variable (or node), and that represent the density for the corresponding branch in the mixed tree. The tree is learnt in such a way that the leaves discriminate as much as possible, following a scheme similar to that carried out by decision trees [Quinlan, 1986]. In order to do so, the following steps must be followed:

1. Selection of the variable to expand from $pa(X)$ by means of the splitting gain.

2. Determination of the splits of the selected variable (for example equal frequency intervals can be used).

3. Learning the MTE. There exists a criterion to stop branching the tree by means of a threshold given by the user.

4. Pruning the tree.

Example 2.1 illustrates a possible conditional MTE density for Y given X (both of them continuous variables).

Example 2.1 *Consider two continuous variables X and Y. A possible conditional MTE density for Y given X is the following:*

$$f(y|x) = \begin{cases} 1.26 - 1.15e^{0.006y} & \textit{if } 0.4 \leq x < 5,\ 0 \leq y < 13\ , \\ 1.18 - 1.16e^{0.0002y} & \textit{if } 0.4 \leq x < 5,\ 13 \leq y < 43\ , \\ 0.07 - 0.03e^{-0.4y} + 0.0001e^{0.0004y} & \textit{if } 5 \leq x < 19,\ 0 \leq y < 5\ , \\ -0.99 + 1.03e^{0.001y} & \textit{if } 5 \leq x < 19,\ 5 \leq y < 43\ . \end{cases}$$

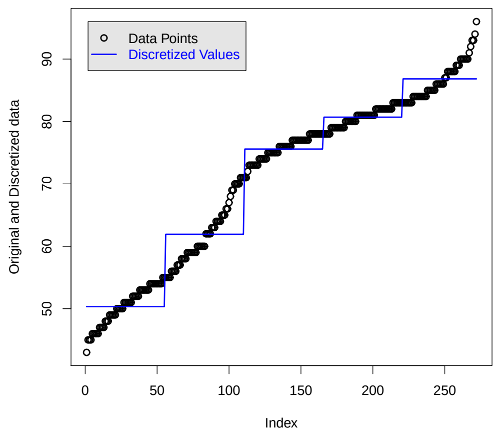
In Langseth *et al.* [2009, 2010] the authors propose an estimation method that directly aims at learning the parameters of an MTE potential following a maximum likelihood approach, instead of existing regression-based methods. Moreover, a model selection scheme is presented based on the Bayesian Information Criterion (BIC) [Schwarz, 1978] for partitioning the domain of the univariate and conditional MTEs.

Most of the work published so far concerning semi-naive Bayesian classifiers with MTEs is mainly focussed on regression [Fernández & Salmerón, 2008b] rather than classification [Flesch *et al.*, 2007], i.e. the class variables to predict are numeric instead of discrete. It makes sense, since MTEs are a good alternative especially in that domain. Nevertheless, the inference mechanisms are similar, and we believe, that the results can also provide an idea on those that would be obtained in the classification domain.

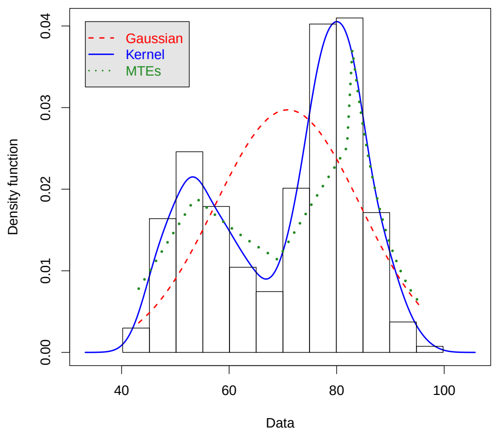
Figure 2.6 shows the graphical results of using the different methods described above to handle a numeric attribute called `waiting`, which represents the waiting time between eruptions for the Old Faithful geyser in the Yellowstone National Park, Wyoming, USA [Azzalini & Bowman, 1990]. The graph on the left hand side shows the average of the values placed in the same bin when applying equal frequency discretization with 5 bins. The graph on the right hand side shows the Gaussian, kernel and MTE estimations.

Another alternative to MTEs is the use of Mixtures of Polynomials (MoPs), proposed in Shenoy & West [2009], where the idea is to substitute the basis function of the MTE, the exponential, by a polynomial. MOP functions are easy to integrate in closed form; and they are closed under multiplication, integration and addition [Giang & Shenoy, 2011; Shenoy, 2011].

In Langseth *et al.* [2012] the authors propose a framework, called mixtures of truncated basis functions (MoTBFs), that generalizes both MTEs and MoPs. It is based on a generalized Fourier series approximation. MoTBFs are claimed to be more flexible than MTEs or MoPs, and support an online/anytime tradeoff between the accuracy and the complexity of the approximation.



(a) Original and discretized data



(b) Gaussian/kernel/MTE estimations

Figure 2.6: Left: The original data points along with the corresponding equal frequency discretization with 5 bins. Right: The histogram of the original data along with: the Gaussian estimate, the kernel estimate and the estimated function using mixtures of truncated exponentials.

2.3 Domains of competence of BNCs in the complexity measurement space

In this section, the reader will find a review on the use of complexity measures specifically designed to define the domain of competence of a particular classifier.

The motivation to resort to this type of measures in this thesis is clear: in order to compare the aforementioned classifiers, we are carrying out empirical studies in a moderate group of datasets so that it is possible to find out, based on error/accuracy rates, the success or failure of a particular classification approach. In Part IV of this thesis, we test to what degree these studies can be enriched by an analysis of classifier's performances based on data characteristics, both for continuous and discrete datasets.

2.3.1 Background

The study of performance of different classifiers is not a recent task in machine learning. Several theoretical and many more empirical studies [Lim *et al.*, 2000; Toh, 2008; Wolpert, 1996] have been carried out. The former attempt to analyse classifier's behaviour for all possible problems and result inevitably in very weak performance bounds, whereas the latter often conclude with a presentation of error rates on a small selection of problems, with little analysis on the reasons behind the classifier's success or failure.

Revealing enough is the work by Jaeger [2003], where the expressivity of classifiers on the different levels in the hierarchy of probabilistic classifiers is characterized algebraically by separability with polynomials of different degrees. The results implies, for the first time, that the concepts recognizable by a naive Bayesian classifier are exactly the linearly separable sets for example.

With the increasing popularity of machine learning techniques, it is becoming more and more interesting to find out a priori, which specific technique will perform better for a particular dataset based on the geometrical characteristics of this dataset. This kind of studies started to receive attention with Sohn [1999], and has become more popular from the work of Ho [2001]; Ho & Basu [2000]. In these studies, the authors indicate the importance in considering detailed descriptions of geometrical characteristics of data, to distribute problems in a measurement space according to its difficulty, so that it is possible to describe a classifier's domain of competence.

This idea matures in Ho & Basu [2002], where a selection of several measures for characterizing the complexity of classification problems is presented, along with an empirical study on the distribution of real world problems compared to random noise, indicating that it is possible to find learnable structures with the geometrical measures presented. These measures indicate the overlap of individual feature values; the separability of classes; and geometry, topology and density of manifolds. We will describe some of them in more detail in Section 2.3.2. This group of measures encounters its natural definition in the two-class domain. Nevertheless, attempts to generalize some of these measures to the multi-class domain can be found in Mollineda *et al.* [2005] and more recently in Orriols-Puig *et al.* [2010].

Numerous studies have followed that try to obtain the domains of competence for one or more particular classifiers, by studying error rate patterns with respect to individual or combination of complexity measures, usually bivariate combinations. Some of these works are Bernadó-Mansilla & Ho [2004, 2005] for 1-nearest-neighbour (1NN), linear classification through linear programming, decision trees, decision forests and XCS; Sánchez *et al.* [2007] for kNN classifier; and more recently, Luengo & Herrera [2009] for artificial neural networks, Luengo & Herrera [2010a] for fuzzy rule based classification systems and Luengo & Herrera [2010c] for C4.5.

In all these papers, experiments have been carried out with a common test bed of datasets where similarity between the datasets for examples is often unknown. In Macià *et al.* [2010], the authors design a procedure to provide problems with a good coverage of the data complexity space to serve as a more complete test bed on the occasion of the ICPR'10 contest "Classifier domains of competence: The Landscape Contest"[1].

Another interesting work in relation to this topic is presented in Hernández-Reyes *et al.* [2005], where an automatic classifier selection based on data complexity measures is proposed. Their method describes problems with complexity measures and labels them with the classifier that gets the best accuracy among a set of five classifiers: kNN, NB, linear regression, RBFNetwork and J48.

The uses of complexity measures are expanding lately. In Miranda [2011], a system of data complexity measures specifically tailored to be employed as predictive attributes in meta-learning for instance selection is presented.

Furthermore, a data complexity library in C++ has been released [Orriols-Puig *et al.*, 2010] that allows to calculate several complexity measures for any database with nominal and/or continuous attributes. Some of these measures can also be found in KEEL, a Java software tool to assess evolutionary algorithms for data mining problems [Alcalá-Fdez *et al.*, 2011].

[1]http://www.salleurl.edu/ICPR10Contest/

2.3.2 Complexity measures

We are including below the different complexity measures for continuous and discrete features as specified in the *data complexity library* (DCoL) [Orriols-Puig *et al.*, 2010], most of them originally proposed in Ho & Basu [2002]. Here, the different measures are divided in 3 groups based on the complexity aspect they focus on: either overlaps in feature values from different classes; separability of classes; and geometry, topology and density of manifolds. In addition, the names of the complexity measures are identified by the letter `L` if it is a linear classifier based, `N` if it is a nearest-neighbour based or `F` if it is a geometry or topology-based measure.

We exclusively focus on two-class datasets, as applying these measures to multi-class problems may hinder some key observations on the complexity related to individual classes.

2.3.2.1 Measures of overlaps in the feature values from different classes

These measures focus on the discriminant power of a single attribute or a combination of them to separate the different classes. They study the range and spread of their values in instances of different classes to check for overlaps among different classes.

F1 - Maximum Fisher's discriminant ratio: It is computed as the maximum of the individual discriminative powers of the different attributes, that is:

$$F1 = \max_{i=1}^{n} \frac{(\mu^i_{c_1} - \mu^i_{c_2})^2}{(\sigma^i_{c_1})^2 + (\sigma^i_{c_2})^2}, \tag{2.18}$$

where; for continuous attributes, $\mu^i_{c_j}$ and $(\sigma^i_{c_j})^2$ are the mean and variance of the attribute A_i for class c_j. For nominal attributes, each value is mapped into an integer number. Then, $\mu^i_{c_j}$ is the median value of the attribute A_i for class c_j, and $(\sigma^i_{c_j})^2$ is the variance of A_i for c_j computed as the variance of the binomial distribution, that is:

$$(\sigma^i_{c_j})^2 = \sqrt{p_{\mu^i_{c_j}}(1 - p_{\mu^i_{c_j}}) \cdot e_{c_j}}, \tag{2.19}$$

where $p_{\mu^i_{c_j}}$ is the frequency of the median value $\mu^i_{c_j}$, and e_{c_j} is the total number of examples of class c_j.

For a multidimensional problem, not all features have to contribute to class discrimination, the problem is easy as long as there exists one discriminating feature. High values of F1 indicate that, at least, one of the attributes enables the learner to separate the examples of different classes with partitions that are parallel to an axis of the feature space. Low values do not imply that the classes are not linearly separable, but that they cannot be discriminated by hyperplanes parallel to one of the axis of the feature space. Fisher's discriminant ratio is good for indicating the separation between two classes each following a Gaussian distribution, but not for two classes forming non-overlapping concentric rings one inside the other.

F1v - Directional-vector maximum Fisher's discriminant ratio: complements F1 by searching for an oriented vector which can separate examples of two different classes. It calculates the two-class Fisher's criterion [Malina, 2001].

A high value of F1v indicates that there exits a vector that can separate examples belonging to different classes after these instances are projected on it.

F2 - Overlap of the per-class bounding boxes: It computes the overlap of the tails of distributions defined by the instances of each class. For each attribute, it computes the ratio between the width of the overlap interval and the width of the entire interval encompassing the two classes. Then, the measure returns the product of per-feature overlap ratios:

$$F2 = \prod_{i=1}^{n} \frac{MIN\_MAX_i - MAX\_MIN_i}{MAX\_MAX_i - MIN\_MIN_i}, \tag{2.20}$$

where:

$$MIN\_MAX_i = \min(\max(A_i, c_1), \max(A_i, c_2)), \tag{2.21}$$

$$MAX\_MIN_i = \max(\min(A_i, c_1), \min(A_i, c_2)), \tag{2.22}$$

$$MAX\_MAX_i = \max(\max(A_i, c_1), \max(A_i, c_2)), and \tag{2.23}$$

$$MIN\_MIN_i = \min(\min(A_i, c_1), \min(A_i, c_2)). \tag{2.24}$$

Again, nominal values are mapped to integer values to compute this measure.

A low value of this measure means that the attributes can discriminate the examples of different classes. It is zero as long as there is at least one dimension in which the value ranges of the two classes are disjoint.

F3 - Maximum (individual) feature efficiency: This measure computes the largest fraction of points distinguishable with only one feature. To this aim, it takes into account for each attribute the region where there are instances of both classes, returning the ratio of the number of instances that are not in this overlapping region to the total number of instances.

A classification problem is easy if there exists one attribute for which the ranges of the values spanned by each class do not overlap (in this case, this would be a linearly separable problem).

F4 - Collective feature efficiency: This measure is similar to F3, but now it considers the discriminative power of all the attributes (therefore, the *collective* feature efficiency).

To compute it, the attribute that can separate a major number of instances of one class is selected. Then, all the instances that can be discriminated are removed from the dataset, and the following most discriminative attribute (with respect to the remaining examples) is selected. This procedure is repeated until all the examples are discriminated or all the attributes in the feature space are considered. Finally, the measure returns the proportion of instances that have been discriminated. Thus, it gives an idea of the fraction of examples whose class could be correctly predicted by building separating hyperplanes that are parallel to one of the axis in the feature space.

The difference with respect to F3, is that the former only considers the number of examples discriminated by the most discriminative attribute, instead of all the attributes. Hence, F4 provides more information by taking into account all the attributes.

2.3.2.2 Measures of class separability

These measures study the separability of the classes by examining the existence and shape of the class boundary. The contributions of individual feature dimensions are combined and *summarized* in a single score, usually a distance metric, rather than evaluated separately.

L1 - Minimized sum of the error distance of a linear classifier: This measure evaluates to what extent the training data is linearly separable. It returns the sum of the difference between the prediction of a linear classifier and the actual class value. A support vector machine (SVM) [Vapnik, 1995] with a linear kernel is used, which is trained with the sequential minimal optimization (SMO) algorithm [Platt, 1999] to build the linear classifier. This learner is selected by Orriols-Puig *et al.* [2010], unlike in Ho & Basu [2002], because the SMO algorithm provides an efficient training method, and the result is a linear classifier that separates the instances of two classes by means of a hyperplane.

A zero value of this measure indicates that the problem is linearly separable.

L2 - Training error of a linear classifier: This measure also provides information about to what extent the training data is linearly separable by returning the training error of the linear classifier as explained above.

It is measured on the training set and when the latter is small, L2 can be a severe underestimate of the true error rate.

N1 - Fraction of points on the class boundary: It gives an estimate of the length of the class boundary by constructing a class-blind minimum spanning tree over the entire dataset, and returning the ratio of the number of nodes of the spanning tree that are connected and belong to different classes to the total number of examples in the dataset.

High values of this measure indicate that the majority of the points lay closely to the class boundary, and hence, that it may be more difficult for the learner to define this class boundary accurately. However, the same can be true for a sparsely sampled linearly separable problem with margins narrower than the distances between points of the same class.

N2 - Ratio of average intra/inter class nearest neighbour distance: This measure compares the within-class spread with the size of the gap between classes. For each input instance e_i, the distance to its nearest neighbour within the class (intraDist(e_i)) and the distance to its nearest neighbour of any other class (interDist(e_i)) are calculated. Then, the result is the ratio of the sum of the intra-class distances to the sum of the inter-class distances for each input example, i.e.,

$$N2 = \frac{\sum_{i=0}^{m} intraDist(e_i)}{\sum_{i=0}^{m} interDist(e_i)}. \tag{2.25}$$

Low values of this measure suggest that the examples of the same class lay closely in the feature space. High values indicate that the examples of the same class are disperse.

N3 - The leave-one-out error rate of the one-nearest neighbour classifier: The measure denotes how close the examples of different classes are. It is simply the leave-one-out error rate of the one-nearest neighbour (1NN) on the training set.

Low values indicate a large gap in the class boundary.

2.3.2.3 Measures of geometry, topology, and density of manifolds

These measures handle an indirect characterization of the class separability. They study the shape, position and interconnectedness of the manifolds that form each class; indicating how well two classes are separated.

L3 - Non-linearity of a linear classifier: This measure implements a measure of non-linearity originally proposed by Hoekstra & Duin [1996]. The method creates a test set from the training dataset, by linear interpolation with random coefficients between pairs of randomly selected instances of the same class. Then, the measure returns the test error rate of the linear classifier (SVM with linear kernel) trained with the original dataset. The measure is sensitive to the smoothness of the classifier's boundary and also to the overlap of the convex hull of the classes. For linear classifiers and linearly separable problems, it measures the alignment of the decision surface with the class boundary. It carries the effects of the training procedure in addition to those of the class separation.

N4 - Non-linearity of the one-nearest neighbour classifier: This measure is exactly as L3 but considering a 1NN instead. It shows the alignment of the NN boundary with the shape of the gap or overlap between the convex hulls of the classes.

T1 - Fraction of maximum covering spheres: This measure uses the notion of *adherence subsets* in pre-topology to describe the shapes of class manifolds [Lebourgeois & Emptoz, 1996]. It counts the number of balls necessary to cover each class, centring each ball at a training point and growing to the maximal size before it touches another class. Balls included in others are removed. Finally, the count is normalized by the total number of points. High values of T1 indicate higher complexity since points are covered by balls of small size, i.e., points are closer to pints of the other class than points of its own class.

T2 - Average number of points per dimension: This measure returns the ratio of the number of instances in the dataset to the number of attributes, i.e.,

$$T2 = m/n.$$

It is a rough indicator of sparseness of the dataset. However, since the volume of a region scales exponentially with the number of attributes, a linear ratio between both is not a good measure of sampling density.

Part II

New BNCs to overcome AODE's limitations

Chapter 3

Hidden one-dependence estimator

We are drowning in information but starved for knowledge.
John Naisbitt. (1929-)
American author

Abstract

In this chapter, a new classifier called hidden one-dependence estimator (HODE) will be presented. It aims to tackle some of the drawbacks that are inherent to AODE's original definition. Hence, the goal is to solve, in the first place, the need to store all the models constructed, that leads to a relatively high demand on space and therefore, to the impossibility of dealing with some problems of high dimensionality; and secondly, reducing the computational time required in classification time (quadratic in the number of attributes), as it is frequently carried out in real time. HODE estimates a new variable (the hidden variable) as a superparent besides the class, whose main objective is to gather the significant dependences existing in AODE models. The obtained results show that this new algorithm provides similar results in terms of accuracy than AODE, with a reduction in space complexity and classification time, and the possibility to be parallelizable.

3.1 Introduction

Even though AODE offers an attractive trade-off between performance and model complexity, it is subject to improvement. Just as discussed above, one of the main drawbacks of AODE is the high space cost it entails, as it is necessary to store all the SPODE models in main memory, $\mathcal{O}(c(nv)^2)$. This is especially problematic when the number of attributes and/or the number of values per attribute is very high. Furthermore, AODE's time complexity is quadratic in the number of attributes in classification time, which could entail a problem in real applications where the response time is critical.

The classifier proposed in this chapter makes use of the EM algorithm [Dempster *et al.*, 1977] to estimate a new superparent variable, with the aim to gather the significant dependences between the predictive attributes and overcome the aforementioned AODE's weaknesses. In addition, we will see how this classifier can be easily parallelizable and extensible to impute missing values using the EM algorithm.

We already find other approaches in literature that tends to improve NB's performance by means of the estimation of hidden variables with different procedures, such as Langseth & Nielsen [2006], with a novel algorithm for learning hierarchical naive Bayes models in the context of classification; or Zhang *et al.* [2005], with their proposal of the hidden naive Bayes.

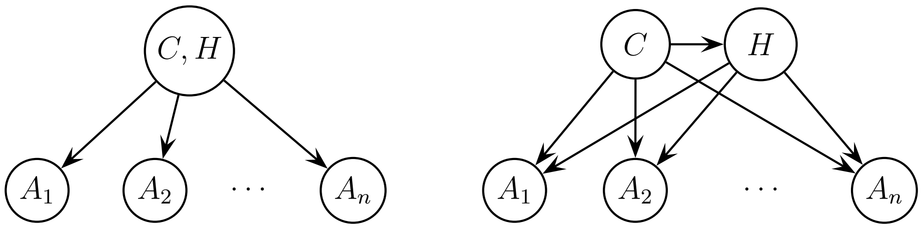
In the following section (Section 3.2), we will describe in detail the new proposed classifier. In Section 3.3, the experimental setup and results are described. Section 3.4 describes the aspects related to parallelization for HODE. Section 3.5 displays an empirical analysis on the imputation of missing values through the EM algorithm in HODE, versus simply ignoring or replacing them by the global mean or mode. It is followed by the main conclusions in Section 3.6.

3.2 HODE classifier

In order to alleviate AODE's large memory requirements, we suggest the estimation of a new variable, specifically, a *hidden variable* H, which gathers the suitable dependences among the different superparents and the rest of the attributes. In other words, instead of averaging the n SPODE classifiers, a new variable is estimated in order to represent the links existing in the n models. This new classifier, as indicated above, will be referred to as HODE.

Definition 3.1 *(HODE classifier [Flores et al., 2009b]) Let $A_1, \dots, A_n$ be a set of features and C a class variable. A HODE classifier is a model that classifies an individual described by features $(a_1, \dots, a_n)$ as belonging to the class c_{MAP} computed as in Equation 3.1, and where all the involved probability functions are obtained through the EM algorithm (as in Algorithm 3.1).*

HODE estimates the probability of every attribute value conditioned on the class and the new variable which plays the superparent role. Figure 3.1 (a) shows the structure of the BN to learn. In our implementation, the class values become the Cartesian product of the original class values and the estimated states for H.



(a) HODE structure (b) HODE alternative structure

Figure 3.1: HODE classifier possible structures.

Equation 3.1 shows the MAP hypothesis for the HODE algorithm. Each h_s, represents the s^{th} virtual value for H, and $\#H$, the final number of states estimated for variable H.

$$c_{MAP} = argmax_{c \in \Omega_C}\ p(c|\vec{e}) = argmax_{c \in \Omega_C} \left(\sum_{s=1}^{\#H} p(c, h_s) \prod_{i=1}^{n} p(a_i|c, h_s) \right). \tag{3.1}$$

Section 3.2.1 illustrates with a very simple example the direct adaptation of the EM algorithm to estimate the probability distributions of the model, whereas Section 3.2.2 explains the technique used to find out the most suitable number of states for H.

3.2.1 Application of the EM algorithm

As the different values for H are not known, we make use of the EM algorithm [Dempster *et al.*, 1977; Gupta & Chen, 2011] to obtain the maximum likelihood estimation of the parameters, its use being quite common in this kind of approaches [Cheeseman & Stutz, 1996; Lowd & Domingos, 2005].

Algorithm 3.1 shows the detailed process. Until convergence is reached, in the Maximization step (M-step) the CPTs are constructed using the weights estimated in the Expectation step (E-step). These weights are, in turn, estimated according to the attribute values, the class value and the corresponding label assigned to H.

Algorithm 3.1: EM algorithm's adaptation to HODE

```
Input: Dataset with variables $A_1, \ldots, A_n, C, \#H$.
Output: Last updated probabilities.
Random initialization of weights.
begin
    while !$convergence()$ do
        //*M-STEP*//
        Update probabilities according to weights.
        //*E-STEP*//
        for $j \leftarrow 1$ to $j = m$ do
            for $(s \leftarrow 0$ to $s < \#H)$ do
                $w_{\{c,h_s,a_i,\cdots,a_n\}_j} = P(c,h_s)P(a_1|c,h_s)\cdots P(a_n|c,h_s)$
            end
            Normalize $w_{\{c,h,a_i,\cdots,a_n\}_j}$
        end
    end
end
return Last updated probabilities
```

In EM the database is virtually divided according to the following procedure: we divide every instance into $\#H$ virtual instances. Each one of the subinstances corresponds to a different value of H and a weight reflecting its likelihood ($w_{\{c,h_s,a_i,\cdots,a_n\}_j}$, for the j^{th} instance with value h_s). At the beginning, these weights are randomly initialized ($\vec{w}$ vector), considering that the sum of weights from a common instance has to be equal to 1. Table 3.1

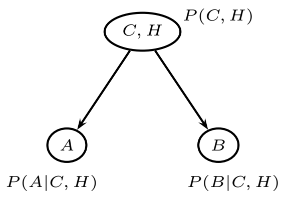
shows a virtual division example for a toy database.

Table 3.1: Virtual division example of a toy database with $H = \{h_1, h_2\}$ in HODE.

A	B	C	H	w
a	b	c	h_1	0.3
			h_2	0.7
a	$\overline{b}$	$\overline{c}$	h_1	0.5
			h_2	0.5
$\overline{a}$	$\overline{b}$	c	h_1	0.9
			h_2	0.1
a	b	c	h_1	0.6
			h_2	0.4
$\overline{a}$	b	$\overline{c}$	h_1	0.7
			h_2	0.3
a	$\overline{b}$	c	h_1	0.2
			h_2	0.8

An example of how to carry out both the E and M steps is described below. For the database in Table 3.1, the probabilities shown in Figure 3.2 are obtained in every M-step.

Structure



A priori probabilities

$$p(c,h_1) = \frac{0.3+0.9+0.6+0.2}{6} = 0.33 \quad p(c,h_2) = \frac{0.7+0.1+0.4+0.8}{6} = 0.33$$

$$p(\overline{c},h_1) = \frac{0.5+0.7}{6} = 0.2 \quad p(\overline{c},h_2) = \frac{0.5+0.3}{6} = 0.13$$

CPT for attributes A and B

$$p(a|c,h_1) = \frac{0.3+0.6+0.2}{2} = 0.55 \quad p(a|\overline{c},h_1) = \frac{0.5}{1.2} = 0.42 \quad p(b|c,h_1) = 0.45 \quad p(b|\overline{c},h_1) = 0.58$$

$$p(a|c,h_2) = \frac{0.7+0.4+0.8}{2} = 0.95 \quad p(a|\overline{c},h_2) = \frac{0.5}{0.8} = 0.625 \quad p(b|c,h_2) = 0.55 \quad p(b|\overline{c},h_2) = 0.375$$

Figure 3.2: Count of database weights to obtain the CPTs in HODE (M-step).

In the E-step, the estimation of the corresponding weights of the virtual instances from the probabilities estimated in the previous step is performed. Equations in Figure 3.3 show how the E-step is carried out for the first instance in our example. Once this E-step is finished for all the instances, the following generation of weights is depicted on the right-hand side table in Figure 3.3.

Finally, the following M-step would use the $\vec{w_2}$ vector weight and the cycle would continue until the algorithm converges, in other words, until the weight differences for all instances, from adjacent iterations, are lower than 5 thousandths.

Weights count

$$p(c, h_1|a, b) = \frac{p(c, h_1)p(a|c, h_1)p(b|c, h_1)}{\sum_{i=1}^{H} (p(c, h_i)p(a|c, h_i)p(b|c, h_i))} = \frac{0.33 \cdot 0.55 \cdot 0.45}{0.254} = 0.32 \tag{3.2}$$

$$p(c, h_2|a, b) = \frac{p(c, h_2)p(a|c, h_2)p(b|c, h_2)}{\sum_{i=1}^{H} (p(c, h_i)p(a|c, h_i)p(b|c, h_i))} = \frac{0.33 \cdot 0.95 \cdot 0.55}{0.254} = 0.68 \tag{3.3}$$

Weights modification after E-step

A	B	C	H	w_1	w_2
a	b	c	h_1	0.3	0.32
			h_2	0.7	0.68
a	$\overline{b}$	$\overline{c}$	h_1	0.5	0.41
			h_2	0.5	0.59
$\overline{a}$	$\overline{b}$	c	h_1	0.9	0.92
			h_2	0.1	0.08
a	b	c	h_1	0.6	0.32
			h_2	0.4	0.68
$\overline{a}$	b	$\overline{c}$	h_1	0.7	0.79
			h_2	0.3	0.21
a	$\overline{b}$	c	h_1	0.2	0.41
			h_2	0.8	0.59

Figure 3.3: Count of database weights in HODE (E-step).

3.2.2 Number of states for the hidden variable

Even though the graphical structure is already fixed, we still have to perform certain learning in order to find the inner structure of H, in other words, its cardinality or number of states. To achieve this, we make use of the following greedy technique: firstly, $\#H$ is fixed to 1 (base case equivalent to naive Bayes), the EM algorithm is executed and the model built is evaluated; after that, the number of states for H is increased one by one in every iteration of the EM algorithm. If the result of the evaluation of one model is better than the one in the previous iteration, the process continues, otherwise, the previous model is restored and considered the final model.

The log-likelihood (LL) measure is used to evaluate the fitness of the model. It calculates how the estimated mathematical model fits the training data. Equation 3.4 shows the formula we have used.

$$LL = \sum_{i=1}^{m} \log \left(\sum_{s=1}^{\#H} p(c^i, a_1^i, \cdots, a_n^i, h_s) \right) = \sum_{i=1}^{m} \log \left(\sum_{s=1}^{\#H} p(c^i, h_s) \prod_{r=1}^{n} p(a_r^i | c^i, h_s) \right), \tag{3.4}$$

where the superscript i indicates the class or the attribute value that correspond with the i^{th} instance.

Nevertheless, when we use these measures, it is also necessary to add another quality measure to counteract the monotonous feature of LL. In other words, it is necessary to somehow penalize the increase in the number of states for H. There are several options in order to achieve this, one of them is the use of the Minimum Description Length (MDL) measure [Rissanen, 1978], for which the model complexity, $C(M)$, is computed as in Equation 3.5.

$$C(M) = \sum_{i=1}^{n} ((\#H \cdot \#C)(\#A_i - 1)) + \#H \cdot \#C - 1\,, \tag{3.5}$$

where $\#C$ is the number of classes and $\#A_i$ the number of states of the attribute A_i.

Thus, the MDL measure can be defined as in Equation 3.6:

$$MDL = LL - \frac{1}{2} \log m \cdot C(M)\,. \tag{3.6}$$

Another way of penalizing LL consists in using information measures, with the basic idea of selecting the model which best fits the data, penalizing according to the number of parameters needed to specify its corresponding probability distribution. Specifically, we are testing the so-called **Akaike Information Criterion** or **AIC** [Akaike, 1978], which turns out to be equal to the previous one but removing the $\frac{1}{2}\log m$ factor.

$$AIC = LL - C(M)\,. \tag{3.7}$$

From our experiments testing these two measures, AIC is the one that provides a smoother penalization over LL and hence, it achieves better results as it explores more states of H (this is in concordance with Lowd & Domingos [2005], where large cardinalities are used in order to achieve good modelling). The graph in Figure 3.4 shows the comparison between accuracy results using both penalty measures. These experiments have been carried out on 36 UCI repository datasets [WEKA-Datasets], whose main characteristics are summarized in Table 3.2 and correspond to the X-axis in the graph. The left-hand Y-axis represents accuracy results (upper pair of lines) whereas the right-hand Y-axis represents the average $\#H$ obtained in the evaluation of each dataset (lower pair of lines). From now on in this chapter, we will simply refer to HODE with AIC as HODE.

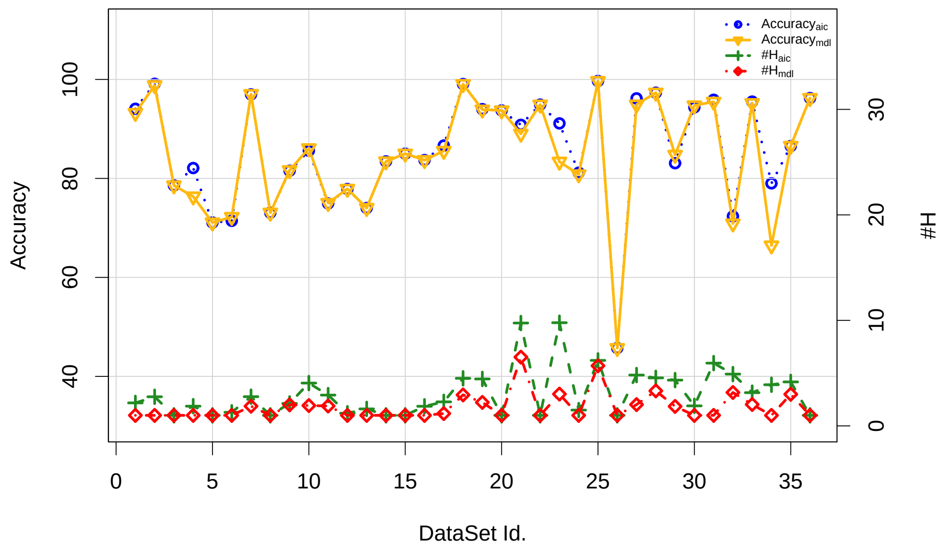


Figure 3.4: Accuracy and $\#H$ obtained with AIC and MDL penalization in HODE.

3.3 Experimental methodology and results

This section presents some experimental results for HODE compared to AODE.

Table 3.2: Main characteristics of the datasets: number of class labels (c), number of predictive variables (n), and number of instances (m).

Id.	Dataset	c	n	m	Id.	Dataset	c	n	m
1	anneal.ORIG	6	38	898	19	ionosphere	2	34	351
2	anneal	6	38	898	20	iris	3	4	150
3	audiology	24	69	226	21	kr-vs-kp	2	36	3196
4	autos	7	25	205	22	labor	2	16	57
5	balance-scale	3	4	625	23	letter	26	16	20000
6	breast-cancer	2	9	286	24	lymph	4	18	148
7	breast-w	2	9	699	25	mushroom	2	22	8124
8	colic.ORIG	2	27	368	26	primary-tumor	21	17	339
9	colic	2	27	368	27	segment	7	19	2310
10	credit-a	2	15	690	28	sick	2	29	3772
11	credit-g	2	20	1000	29	sonar	2	60	208
12	diabetes	2	8	768	30	soybean	19	35	638
13	glass	6	10	214	31	splice	3	61	3190
14	heart-c	2	13	303	32	vehicle	4	18	846
15	heart-h	2	13	294	33	vote	2	16	435
16	heart-statlog	2	13	270	34	vowel	11	13	990
17	hepatitis	2	19	155	35	waveform-5000	3	40	5000
18	hypothyroid	4	29	3772	36	zoo	7	17	101

Firstly, in Section 3.3.1 we study the accuracy results obtained on the 36 datasets above mentioned; whereas Section 3.3.2 is devoted to the study of the performance of HODE in terms of efficiency.

We have adopted three pre-processing steps, in order to make the group of datasets suitable for the classifiers considered in the comparison:

- Unsupervised filter to replace all the missing values with the modes and means from the existing data in the corresponding column.
- Supervise filter to discretize the datasets using Fayyad & Irani's MDL method based on minimum entropy [Fayyad & Irani, 1993].
- Unsupervised filter to remove attributes that do not vary at all or whose variance percentage is greater than 99%.

3.3.1 Evaluation in terms of accuracy

Table 3.3 shows the classification accuracy of both classifiers, AODE and HODE, on each dataset obtained via 10 runs of ten-fold cross validation (cv).

Table 3.3: Accuracy results obtained with AODE and HODE classifiers.

Dataset	AODE	HODE	$\overline{\#H}$	Dataset	AODE	HODE	$\overline{\#H}$
anneal.ORIG	93.3185	▲**94.0646**	2.2	ionosphere	92.9915	▲**93.9886**	4.4
anneal	98.1960	▲**99.1203**	2.8	iris	93.2000	▲**93.7333**	1.0
audiology	71.6372	▲**78.5841**	1.0	kr-vs-kp	**91.0325**	90.8229	9.7
autos	81.3658	▲**82.0975**	1.9	labor	**95.0877**	94.9123	1.0
balance-scale	69.3440	▲**71.0880**	1.0	letter	88.9020	▲**91.1170**	9.8
breast-cancer	▲**72.7273**	71.4336	1.3	lymph	▲**87.5000**	81.1487	1.5
breast-w	96.9671	**96.9814**	2.8	mushroom	▲**99.9508**	99.6824	6.2
colic.ORIG	▲**75.9511**	73.0707	1.0	primary-tumor	▲**47.8761**	45.7227	1.0
colic	▲**82.5543**	81.5489	2.1	segment	95.7792	▲**96.1732**	4.8
credit-a	▲**86.5507**	85.5942	4.1	sick	▲**97.3966**	97.3118	4.6
credit-g	▲**76.3300**	74.9400	2.9	sonar	▲**86.5865**	83.0769	4.3
diabetes	▲**78.2292**	77.8516	1.2	soybean	93.3089	▲**94.3631**	1.9
glass	▲**76.2617**	74.0187	1.6	splice	▲**96.1160**	95.8872	3.9
heart-c	83.2013	▲**83.4323**	1.0	vehicle	72.3049	**72.3522**	4.9
heart-h	84.4898	**85.0000**	1.0	vote	94.5288	▲**95.5173**	3.1
heart-statlog	82.7037	▲**83.7037**	1.9	vowel	▲**80.8788**	79.0101	3.9
hepatitis	85.4839	▲**86.6452**	2.3	waveform-5000	86.4540	**86.5400**	4.2
hypothyroid	98.7513	▲**99.0668**	4.5	zoo	94.6535	▲**96.2376**	1.0

Each value represents the arithmetical mean from the 10 executions. The black triangle next to certain outputs means that the corresponding classifier on this particular dataset is significantly better than the other classifier. The results were compared using a two-tailed t-test with a 95% confidence level.

In 16 of the 36 databases, HODE is significantly better than AODE, whereas AODE outperforms HODE in 14 of them. They draw in 6 of them, hence 16/6/14, where the notation $w/t/l$ means that HODE wins in w datasets, ties in t datasets, and loses in l datasets, compared to AODE. The results undergo no variation when the confidence level is raised to 99%, obtaining 15/8/13.

On the other hand, although it is not shown in the tables, we also studied HODE with the MDL penalization, and observed that it was significantly better than AODE in 11 of the 36 datasets, drew in 7 of them, and lost in 18 (11/7/18).

Note that even though we have applied HODE on exclusively discrete datasets for fair comparisons with AODE, it is fitting to point out the fact that HODE is trivially applied on hybrid datasets (that contain continuous and discrete attributes) using Gaussian distributions, similarly to NB.

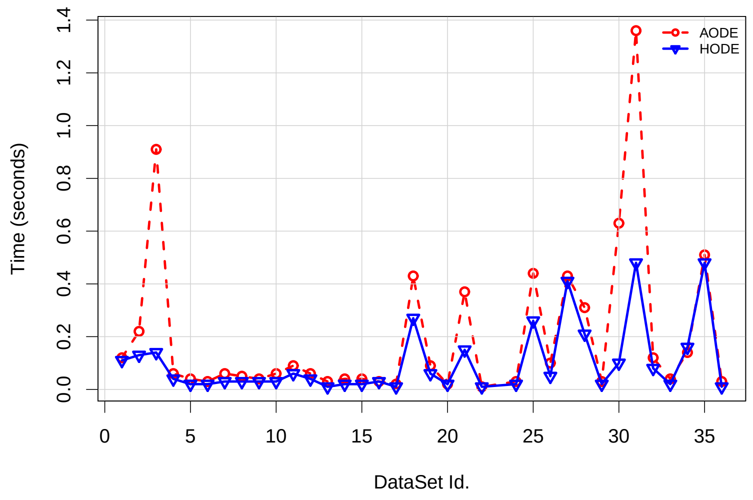


Figure 3.5: Classification time comparison between AODE and HODE.

3.3.2 Evaluation in terms of efficiency

As there is not a clear difference in terms of accuracy between the two classifiers, what could make us vote for one or the other? In fact, HODE's time complexity at *training time* is quadratic in the worst case ($1mn + 2mn + \cdots + nmn$, considering the different executions of the EM algorithm). However, AODE is usually faster than HODE in model construction, as HODE spends more time executing the EM algorithm to find the most suitable $\#H$, increasing this time as $\#H$ increases.

With respect to *classification time*, HODE's is linear, whereas AODE's is quadratic. Figure 3.5 shows the experimental classification times obtained, which corroborate this theoretical study. Note that in most real applications, it is essential that classification time is as short as possible, as model training can usually be performed offline. For example, consider spam detection in mail, the recommendation of a specific product according to previous purchases, interpretation of characters by an OCR tool, determining the folder for a certain e-mail, etc.

Furthermore, space complexity for AODE is higher than HODE's, as the former needs to store more CPTs. In fact, HODE's is $\mathcal{O}(n\#Hvc)$, where $\#H$ is usually much lower than n[1]. This requirement in AODE leads to a higher demand on RAM memory, which could be a problem in large databases with a high number of attributes, such as microarrays or DNA chips. To corroborate this fact, we have experimented with a group of 7 databases of this type (see left part of Table 3.4). AODE had problems of overflow with a maximum of 8 gigabytes of memory available, while HODE terminated its executions without problems, even with a lower need for memory.

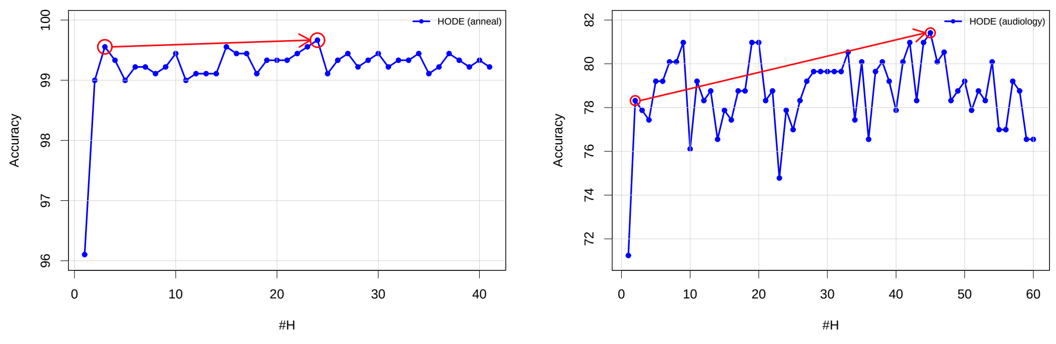
Table 3.4: Main characteristics of the datasets (number of different values of the class variable (c), number of genes (n), and number of microarrays (m)); and accuracy results obtained with NB, AODE and HODE classifiers in these datasets.

Dataset	c	n	m	NB	AODE	HODE
colon	2	2000	2	93.5484	91.9355	**96.7742**
DLBCL-Stanford	2	4026	47	100.0000	100.0000	100.0000
GCM	14	4026	47	60.5263	OutOfMem	**70.0000**
leukemia	2	7129	72	**100.0000**	OutOfMem	98.6111
lungCancerHarvard2	175	12533	181	98.8950	OutOfMem	**99.4475**
lymphoma	9	4026	96	**96.8750**	OutOfMem	75.0000
prostate_tumorVS	2	12600	136	80.1471	OutOfMem	**95.5882**

3.4 HODE's parallelization

HODE can be parallelizable by dividing the time employed in the most costly part indeed. Hence, it is possible to assign the different executions of the EM algorithm to different processors, while exploring the optimal number of states for H. This permits to decrease the factual training time and also the likelihood to obtain better accuracy rates, since it is more likely to find a global value for $\#H$ than through the sequential version. Figure 3.6 (a) shows an acceptable local optimum in terms of trade-off between complexity and

[1] In fact, HODE's exact space complexity is $\theta(2n\#Hvc)$, as there is a need to store a *backup* set of CPTs while performing EM. However, if $m < nvc$, we could consider to store directly the weights estimated in the E-step, hence, obtaining a complexity $\theta(n\#Hvc + m\#H)$ instead.



(a) anneal (b) audiology

Figure 3.6: Differences between local and global optimum for `anneal` and `audiology` datasets with HODE.

performance, whereas Figure 3.6 (b) displays a less desirable local optimum, which is further from the global one or even from better local optimums.

Even though parallelization entails an extra cost in communicating the results obtained by each process, experiments with the mpiJava library [Baker *et al.*, 1999] show that this overload can be dismissed for even datasets of relatively small size, such as `soybean` [Michalski & Chilausky, 1980]. In Figure 3.7, the behaviour in terms of time, depending on the number of processors for three toy datasets: `weather`, `labor` and `soybean`, is displayed. Whereas weather is shown to be too small to be parallelized, labor's training time is reduced by using 2 processors, and soybean even using three processors. Note that, the larger the dataset, the bigger the speedup obtained by using a larger number of processors, which is a desirable property in data mining.

3.5 Analysis of missing values in AODE and HODE

As HODE makes use of the EM algorithm to estimate the probability distributions, it is logical to think of using this algorithm to take advantage of its application for simultaneously carrying out an imputation of the missing

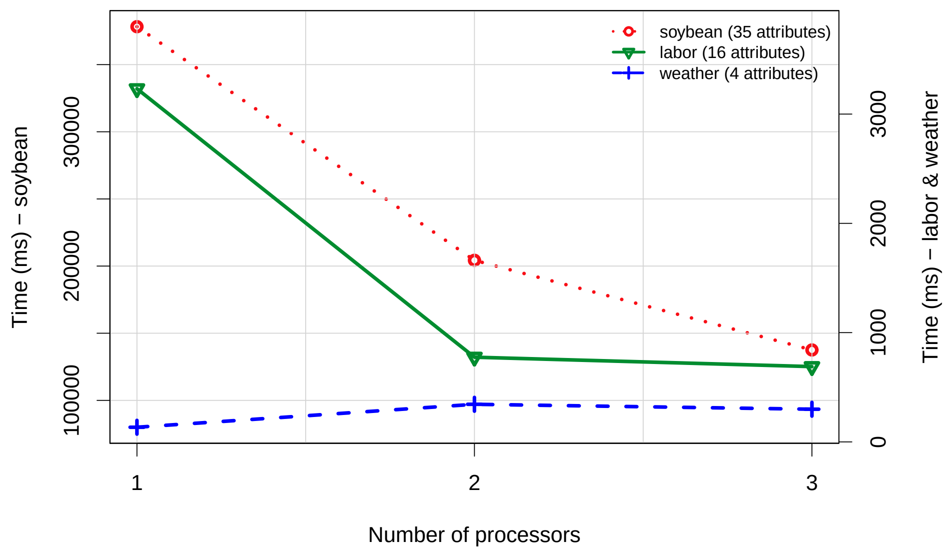


Figure 3.7: Time employed by the parallelized version of HODE to train three particular datasets: `weather`, `labor` (right Y-axis) and `soybean` (left Y-axis), when 1, 2 or 3 processors are used.

values in data. In this empirical study, we are reproducing the same conditions as in Section 3.3, and we are analysing the performance obtained when the presence of missing values is faced in these three forms:

1. Missing values are ignored by the classification method.
2. Missing values are included in the EM algorithm in HODE.
3. Missing values are "a priori" imputed and replaced by the mean and mode of the attribute.

Subsection 3.5.1 contains the comparisons between the first and second methodology, and Subsection 3.5.2 introduces the results when the third methodology is applied.

We do not consider more sophisticated techniques in order to maintain the simplicity and efficiency desired in HODE. For the experiments, we take the datasets with missing values out of the group of datasets in Table 3.2, they are summarized in Table 3.5 in increasing order of missing values percentage.

Table 3.5: Main characteristics of the datasets with missing values: column $\%M$ indicates the percentage of missing values (increasing order).

Id.	Dataset	n	c	I	$\%M$
1	breast-w	38	6	898	0.23
2	breast-cancer	38	6	898	0.32
3	mushroom	38	6	898	1.33
4	audiology	69	24	226	2.00
5	heart-c	13	2	303	2.42
6	primary-tumor	69	24	226	2.00
7	credit-a	15	2	690	5.00
8	vote	16	2	435	5.30
9	hepatitis	19	2	155	5.39
10	hypothyroid	29	4	3772	5.40
11	sick	29	2	3772	5.40
12	soybean	35	19	638	9.51
13	autos	25	7	205	11.06
14	colic.ORIG	27	2	368	18.70
15	heart-h	13	2	294	19.00
16	colic	27	2	368	22.77
17	labor	16	2	57	33.64
18	anneal.ORIG	38	6	898	63.32

3.5.1 Missing values ignored vs included in the EM algorithm

Table 3.6 shows the accuracy results for AODE, HODE ignoring missing values, and HODE using EM in order to impute them (HODEMissing). It does not seem to be worthy, at least for this group of datasets, to give a special treatment for missing values inside the EM algorithm. HODEMissing wins in 7 datasets vs 11 for HODE. The number of H states selected in both cases is very similar, still, estimations made by HODE ignoring the missing values seem to be slightly more accurate.

To be in a better position to draw conclusions, we perform a different experiment. Now, we take several datasets and start to include missing values at random at increasing percentages. The results are displayed in Table 3.7, where the first column $\%M$ indicates the increasing percentage of missing values in every case. With this study, we do not obtain a clear

Table 3.6: Accuracy results for AODE, HODE (ignoring missing values) and HODEMissing. The bullet next to certain outputs indicates an improvement on the comparison of this value with the corresponding results in AODE.

Id	AODE	HODE	estH	HODEMissing	estH
1	97.1674	•97.0815	2.76	97.0672	2.62
2	73.0420	70.8741	1.32	•71.8881	1.27
3	99.9508	99.6984	6.17	•99.8104	6.68
4	72.6991	•78.7611	1.00	78.0088	1.00
5	83.3663	83.6964	1.00	•83.7624	1.00
6	49.7640	47.4041	1.00	•47.8761	1.00
7	86.2029	85.5507	4.08	•86.0870	3.83
8	94.2759	•95.9540	3.18	95.8161	3.04
9	86.6452	•86.9677	2.22	86.4516	2.21
10	99.0376	•99.2550	4.46	99.2391	4.15
11	97.3648	•97.3568	4.45	97.3515	4.83
12	93.2064	•94.6706	1.00	94.0703	1.05
13	82.5854	83.5610	1.89	•84.0488	1.93
14	71.9022	•71.9293	1.00	70.4348	1.00
15	84.2857	•84.9660	1.00	83.7755	1.00
16	83.3967	81.5217	2.40	•82.7717	2.25
17	92.4561	•92.1053	1.00	91.9298	1.00
18	97.2272	•97.3942	1.92	96.0690	1.96
Av.	85.8098	**86.0416**	2.32	85.9143	2.32

pattern either, as for some datasets using EM to impute these missing values seems to be beneficial in most cases (`autos` in Table 3.7), whereas it is not for others (`soybean` in Table 3.7).

3.5.2 Missing values imputed with the global mean/mode

Table 3.8 displays the results both for AODE and HODE, when ignoring the missing values or imputing these values using the global mean/mode for each attribute (`Imputed G.`). The bullet next to certain outputs in these columns, indicates an improvement on the comparison of this value with the corresponding results when ignoring the missing values (columns `Ignored` in AODE or HODE); the circle indicates a tie.

The conclusions we can obtain in the light of the results are the following:

- Performing the imputation of missing values with the EM algorithm does not seem to be overall beneficial (total average accuracy), although HODEmissing provides better records (12-0-6) than imputation with

Table 3.7: Accuracy results for HODE and HODEMissing when missing values at random at increasing percentages are included in datasets `autos`, `labor` and `soybean`. The bullet next to certain outputs indicates an improvement on the comparison of this value (in HODEMissing) with the corresponding results in HODE; whereas the circle indicates a tie.

	autos		labor		soybean	
$\%M$	HODE	HODEMissing	HODE	HODEMissing	HODE	HODEMissing
0	83.0244	◦83.0244	94.3860	94.3860	94.3777	94.3777
10	76.8293	•79.6098	94.2105	•94.3860	94.1288	93.9678
20	73.5610	•74.8780	92.6316	92.6316	93.0161	92.8111
30	70.3902	•71.2195	87.8947	•88.9474	91.5373	88.9898
40	66.9756	•67.6585	87.0175	85.4386	89.6486	86.3836
50	63.4634	•65.3171	84.9123	84.9123	86.1054	83.2943
60	61.5122	•63.1220	87.7193	85.7895	80.8346	77.3939
70	56.0976	•59.1220	84.2105	•84.3860	72.2694	69.0190
80	51.5610	50.9756	73.1579	•75.0877	58.1259	52.5476
90	39.9512	38.5854	68.2456	67.0175	34.0996	30.6003

Table 3.8: Accuracy results for AODE and HODE when imputing missing values with the global mean/mode (Imputed G.). The bullet next to certain outputs indicates an improvement on the comparison of this value with the corresponding results when ignoring the missing values (columns Ignored in AODE or HODE); the circle indicates a tie.

	AODE		HODE		
Id.	Ignored	Imputed G.	Ignored	Imputed G.	HODEMissing
1	97.1674	96.9671	97.0815	96.9814	97.0672
2	73.0420	72.7273	70.8741	•71.4336	•71.8881
3	99.9508	◦99.9508	99.6984	99.6824	•99.8104
4	72.6991	71.6372	78.7611	78.5841	78.0088
5	83.3663	83.2013	83.6964	83.4323	•83.7624
6	49.7640	47.8761	47.4041	45.7227	•47.8761
7	86.2029	•86.5507	85.5507	•85.5942	•86.0870
8	94.2759	•94.5287	95.9540	95.5172	•95.8161
9	86.6452	85.4839	86.9677	86.6452	86.4516
10	99.0376	98.7513	99.2550	99.0668	•99.2391
11	97.3648	•97.3966	97.3568	97.3118	•97.3515
12	93.2064	•93.3089	94.6706	94.3631	94.0703
13	82.5854	81.3659	83.5610	82.0976	•84.0488
14	83.3967	82.5543	81.5217	•81.5489	•82.7717
15	84.2857	•84.4898	84.9660	•85.0000	83.7755
16	71.9022	•75.9511	71.9293	•73.0707	70.4348
17	92.4561	•95.0877	92.1053	•94.9123	91.9298
18	97.2272	93.3185	97.3942	94.0646	•96.0690
Av.	85.8098	85.6193	86.0416	85.8349	85.9143

the global mean/mode.

- HODE seems to be, in nature, a bit more robust against missing values than AODE. When the missing values are ignored, HODE tends to outperform AODE in most of the datasets with higher percentage of missing values (Table 3.8). The reason could be that in AODE these missing values are ignored twice (as children and parents) whereas in HODE only once.

3.6 Conclusions and future work

HODE provides a reduction in space complexity and classification time as well (linear complexity order). The latter leads to a lower time response in many real applications and a lower RAM consumption. Basically, HODE estimates a new variable whose main objective is to model the meaningful dependences between each attribute and the rest of the attributes that AODE takes into account. In order to estimate the number of states of this new variable, we make use of the EM algorithm, evaluating the fitness for every model with a greedy technique.

So far, we have shown empirically how HODE can be considered an attractive alternative to AODE, especially in high dimensional datasets (where the number of attributes, or number of values per attributes is very large), for which it may become the only alternative, since AODE requires larger memory requirements.

Besides, we have introduced the promising performance of HODE in a parallel environment, as we are able to find a global optimum for $\#H$. An additional advantage of HODE would be the direct adaptation to work with missing values in the dataset, due to the use of EM in its main cycle. Furthermore, HODE seems to be more robust against missing values than AODE.

Finally, it would be of a major interest, in order to speed up the training process, to investigate how the estimations on one step in the EM algorithm used in HODE can be reused on posterior steps [Karciauskas, 2005].

Chapter 4

Gaussian AODE and hybrid AODE

The harmony of the universe knows only one musical form - the legato; while the symphony of number knows only its opposite - the staccato. All attempts to reconcile this discrepancy are based on the hope that an accelerated staccato may appear to our senses as a legato.

Tobias Dantzig. (1884 - 1956)

Baltic German Russian American mathematician

Abstract

Within the framework of BNs, most classifiers assume that the variables involved are of a discrete nature, but this assumption rarely holds in real problems. In order to offer an alternative to discretization, in this chapter, we present two different approaches based on Gaussian distributions to deal directly with numeric attributes. One of them uses conditional Gaussian networks to model a dataset exclusively with numeric attributes; and the other one keeps the superparent on each model discrete and uses univariate Gaussians to estimate the probabilities for the numeric attributes and multinomial distributions for the categorical ones, it also being able to model hybrid datasets. Both of them obtain competitive results compared to AODE, the latter in particular being an attractive alternative to AODE in numeric datasets.

4.1 Introduction

The paradigm of BNs assume all random variables are multinomial. Most of the algorithms and procedures designed for Bayesian classifiers are only able to handle discrete variables, so when a numeric variable is present, it must be discretized. In Pérez *et al.* [2006], wrapper and filter approaches are designed to adapt four well-known paradigms of discrete classifiers for handling continuous variables (namely NB, TAN, KDB and the semi naive Bayes using joint variables). However, so far, the only way of training AODE with a dataset containing numeric attributes has been to discretize this dataset before building the model, which can be a handicap in many situations as this process, by definition, entails an inherent loss of information.

Nevertheless, when numerical variables are considered, the problem arises of how to model the probability distribution for a variable conditioned, not only by the class (which is discrete), but also by another numeric attribute. Gaussian networks (GNs) [Geiger & Heckerman, 1994] have been proposed as a good alternative to the direct discretization of continuous attributes. A GN is similar to a BN, but it assumes all attributes are sampled from a Gaussian density distribution, instead of a multinomial distribution. Despite this strong assumption, Gaussian distributions usually provide reasonable approximations to many real-world distributions.

In this chapter, two approaches are proposed to handle continuous variables in AODE: GAODE and HAODE. Both of them inherit the same structure as AODE. In the first one, we make use of CGNs to model the relationship between a numeric attribute conditioned to a discrete class and another numeric attribute; and hence, it is restricted to numerical datasets. In the second one, a discrete version of the superparent attribute is considered in every model, so the previous relationship can be estimated by a univariate Gaussian distribution. The latter approach applies multinomials for nominal children, being able to deal directly with datasets with continuous and/or discrete attributes.

This chapter is organized as follows: Sections 4.2 and 4.3 provide a detailed explanation of the two algorithms designed. In Section 4.4 we describe the experimental setup and results. And finally, Section 4.5 summarizes the main conclusions of our chapter and outlines the future work related to this study.

4.2 Gaussian AODE (GAODE) classifier

Definition 4.1 *(GAODE classifier [Flores et al., 2009a]) Let $A_1, \ldots, A_n$ be a set of continuous features and C a class variable. A GAODE classifier is a model that classifies an individual described by features $(a_1, \ldots, a_n)$ as belonging to the class c_{MAP} computed as in Equation 4.2, and where all the involved probability functions are of conditional Gaussian class (Equation 4.1).*

The underlying idea of this classifier consists in using CGNs to deal with continuous attributes in AODE. In fact, as the class variable is discrete, if we restrict all the predictive attributes to be continuous, we can make use of the Bayes rule to combine Bayesian and Gaussian networks to encode the joint probability distributions among the domain variables, based on the conditional independences defined by AODE.

In the particular case of AODE's structure, the density function for every predictive attribute has to be estimated over a node with a single discrete parent, that is, the class C and another continuous parent, which is the superparent attribute in every model, A_j. The adaptation of the CG density function in Equation 2.11 to this case is:

$$f(A_i = a_i | C = c, A_j = a_j) = \mathcal{N}\left(a_i : \mu_i(c) + b_{ij}(c)(a_j - \mu_j(c)), \sigma^2_{i|j}(c)\right) \quad (4.1)$$

The Bayesian structure for GAODE would remain the same as AODE, and its MAP hypothesis is obtained when we replace the multinomial probability distributions in Equation 2.4 (page 13), with the corresponding CG distribution function defined in Equation 4.1. Whereas the relationship between every predictive attribute conditioned on the class and the corresponding superparent is modelled by a CG distribution, the relationship between every superparent and the class is modelled by a univariate Gaussian distribution. Hence, assuming all the predictive variables are continuous, GAODE selects the class label which maximizes the following summation:

$$argmax_c\left(\sum_{j=1}^{n}\mathcal{N}\left(a_j : \mu_j(c), \sigma_j^2(c)\right)p(c)\right.$$
$$\left.\prod_{i=1\wedge i\neq j}^{n}\mathcal{N}\left(a_i : \mu_i(c) + b_{ij}(c)(a_j - \mu_j(c)), \sigma_{i|j}^2(c)\right)\right) \quad (4.2)$$

More details on CGNs and the calculations of these parameters can be found in Section 2.2.2.

As we can deduce from our definition of this classifier with the application of CGNs, it is not possible to define the corresponding probability function for a discrete variable conditioned on a numeric attribute [Lauritzen & Jensen, 2001]. As in AODE, all the attributes play the superparent role in one model, none of the children attributes are allowed to be discrete and therefore, GAODE is only defined to deal with datasets exclusively formed by numeric attributes (plus, of course, the discrete class).

In this case, the space complexity at *training* and *classification time* becomes independent of the number of values per attribute v, and equals $\mathcal{O}(kn^2)$. Furthermore, as the number of necessary parameters is independent of v, the probabilities estimated can be more reliable compared to the multinomial version as they are modelled from more samples, specially when the size of the CPTs is very large.

The time complexity undergoes no variation as the parameters of the different Gaussian and CG distributions can be computed incrementally.

4.3 Hybrid AODE (HAODE) classifier

Definition 4.2 *(HAODE classifier [Flores et al., 2009a]) Let $A_1, \ldots, A_n$ be a set of (continuous and/or discrete) features and C a class variable. A GAODE classifier is a model that classifies an individual described by features $(a_1, \ldots, a_n)$ as belonging to the class c_{MAP} computed as in Equation 4.3 using* *the discretized version of a_j, and where all the involved probability functions are of Gaussian or multinomial class.*

As we have seen, the GAODE classifier as defined above, is only able to deal with datasets which exclusively contain continuous attributes. In order to include the possibility of handling all kind of datasets, we decide to consider every superparent as discrete in its corresponding model, in principle, by means of any discretization method. However, only the superparent will be discretized, for the rest of attributes their numeric value will be considered. In this way, there is no need to resort to conditional Gaussian distributions, as all the parents in the network will be discrete, but at the same time, we keep most of the original precision from the numeric data.

This can also be seen as an even more simple way of solving the problem of dealing with the continuous superparents on each model in AODE, as there is no need to use conditional Gaussian distributions, but only univariate Gaussians, as in Gaussian NB.

Hence, the MAP hypothesis is developed in the following way:

$$argmax_c\left(\sum_{j=1,N(a_j)>q}^{n} p(a_j,c)\prod_{i=1\wedge i\neq j}^{n}\mathcal{N}\left(a_i:\mu_i(c,a_j),\sigma_i^2(c,a_j)\right)\right) \qquad (4.3)$$

This means that the relationship between the superparent and the class is modelled with a multinomial probability distribution, whereas the rest of relationships, where every other attribute is conditioned on the class and the superparent, are modelled by univariate Gaussian distributions, as long as they are continuous.

As we have pointed out above, this new classifier offers the additional advantage of dealing with datasets that contain a mixture of discrete and continuous variables. In the cases where the child attribute is discrete, a multinomial distribution will be used, as in AODE. This feature represents a significant advantage with respect to the use of CGNs proposed in the previous section, as well as an evident simplification in the calculation of parameters.

The models constructed are 1-dependent, which is why the required CPTs to store the different probability distributions, when necessary for HAODE, are still three-dimensional, as in AODE. In this case, space complexity will increase with the number of discrete variables in the dataset, the top level being the same as for AODE, $\mathcal{O}(c(nv)^2)$.

In both classifiers the model selection between the n SPODE models is unnecessary, as in AODE, thus avoiding the computational cost required by this task and hence maintaining AODE's efficiency and minimizing the variability in the error obtained.

4.4 Experimental methodology and results

4.4.1 Numeric datasets

In order to evaluate the performance of the two classifiers developed, we have carried out experiments over a total of 26 numeric datasets, downloaded from the homepage of the University of Waikato [WEKA-Datasets]. We gathered together all the datasets on this web page, originally from the UCI repository [Frank & Asuncion, 2010], which are aimed at classification problems and exclusively contain numeric attributes according to WEKA [Hall *et al.*, 2009]. Table 4.1 displays these datasets and their main characteristics.

Table 4.1: Main characteristics of the 26 numeric datasets: number of predictive variables (n), number of classes (c) and number of instances (m).

Id	Datasets	n	c	m	Id	Datasets	n	c	m
1	balance-scale	4	3	625	14	mfeat-fourier	76	10	2000
2	breast-w	9	2	699	15	mfeat-karh	64	10	2000
3	diabetes	8	2	768	16	mfeat-morph	6	10	2000
4	ecoli	7	8	336	17	mfeat-zernike	47	10	2000
5	glass	9	7	214	18	optdigits	64	9	5620
6	hayes-roth	4	4	160	19	page-blocks	10	5	5473
7	heart-statlog	13	2	270	20	pendigits	16	9	10992
8	ionosphere	34	2	351	21	segment	19	7	2310
9	iris	4	3	150	22	sonar	60	2	208
10	kdd-JapanV	14	9	9961	23	spambase	57	2	4601
11	letter	16	26	20000	24	vehicle	18	4	946
12	liver-disorders	6	2	345	25	waveform-5000	40	3	5000
13	mfeat-factors	216	10	2000	26	wine	13	3	178

Table 4.2 shows the accuracy results obtained when using 5x2cv[1] to evaluate the different classifiers, as it entails a reasonable trade-off between precision and execution time of the experiments, providing a better partition for the posterior statistical analysis, as in addition, the degree of overlapping between the different folds is lower [Dietterich, 1998]. Each value represents the arithmetical mean from the 10 executions. The black square next to certain outputs means that the corresponding classifier on this particular dataset either obtains the highest accuracy or is not significantly worse than the classifier which does. The results were compared using the 5x2cv F Test defined by Alpaydin [1999], which has lower type I error and higher power than the 5x2cv t-test. The level of significance was fixed at 95% ($\alpha = 0.05$). The 5x2cv F Test is more conservative than the 5x2cv t-test, so a higher number of ties will be obtained with the same level of significance.

Besides GAODE and HAODE, three other classifiers were included in the comparison. From left to right: NB with Gaussian distributions to deal with continuous attributes (GNB); and NB and AODE with the datasets previously discretized using Fayyad and Irani's MDL method [Fayyad & Irani, 1993] (simply identified as NB and AODE). The corresponding discretization of the superparent attributes in HAODE was also carried out using this method[2].

Table 4.3 shows, in the upper half of each cell, the comparison between every pair of algorithms, where each entry w-t-l in row i and column j means that the algorithm in row i wins in w datasets, ties in t (ties means no statistical difference according to the 5x2cv F Test) and loses in l datasets, compared to the algorithm in column j. The lower half of each cell contains the results from the Wilcoxon tests [Demšar, 2006], with $\alpha = 0.05$, which compare every pair of algorithms with the 26 datasets: whenever the test result represented a significant improvement in favour of one of the tests over the other, the name of the winner is shown, otherwise `NO` is shown.

In terms of the arithmetical mean obtained, NB with discretization might

[1] 5x2cv means performing 2-folds cross validation 5 times (randomizing the data).

[2] Further experiments have been performed with different discretization methods, and the results obtained follow the same tendency (further details in Chapter 6).

Table 4.2: Accuracy results obtained for NB with Gaussians (GNB), NB, AODE, GAODE and HAODE in continuous datasets. The black square next to certain outputs means that the corresponding classifier on this particular dataset either obtains the highest accuracy or is not significantly worse than the classifier which does.

Id	GNB	NB	AODE	**GAODE**	**HAODE**
1	■88.8640	77.6320	76.9920	■89.0880	■87.6800
2	■96.0801	■97.1102	■96.6237	■95.9662	■95.0787
3	■74.9740	■74.6875	■74.5573	■74.7917	■75.9115
4	■83.9881	■80.7738	■81.0119	■84.5238	■84.3452
5	49.7196	■60.0000	■60.7477	■52.8037	■60.6542
6	■65.3750	■57.5000	■57.5000	■65.6250	■68.5000
7	■83.4815	■81.2593	■80.8148	■83.7778	■83.0370
8	■82.9630	■88.8889	■90.7123	■92.0228	■91.7379
9	■95.0667	■93.4667	■93.3333	■97.4667	■95.6000
10	85.7444	84.5758	90.3885	91.8442	■93.9966
11	64.0600	73.2960	■86.2920	71.2350	■86.1380
12	■54.2609	■58.6087	■58.6087	■57.3333	■54.2029
13	92.2900	92.3600	■96.0800	■95.9400	■96.3100
14	75.7000	75.8700	79.2500	■79.3900	■80.6900
15	93.1600	90.4800	■93.8300	■96.1500	■95.9200
16	■69.3200	68.0300	68.9000	■70.7900	■69.9500
17	72.9900	70.2100	74.6300	■77.4200	■78.1000
18	91.1317	91.7544	■96.3167	93.6370	■96.9181
19	■87.7142	93.1336	■96.6307	■90.9446	■91.8144
20	85.7041	87.3362	■97.1161	94.2085	■97.5182
21	80.6753	90.4416	■94.1732	86.6667	■95.1602
22	67.5000	■75.6731	■75.5769	■71.4423	■75.9615
23	79.5131	89.8544	■92.7277	79.8566	77.3658
24	43.1678	58.6052	67.4704	■68.5106	■72.9787
25	80.0000	79.9680	■84.5080	■84.4600	■84.2200
26	97.4157	96.9663	■96.9663	■98.4270	97.4157
Av	78.4842	80.3262	83.1445	82.4739	84.1233

Table 4.3: Accuracy comparison between pairs of algorithms: GNB, NB, AODE, GAODE and HAODE.

Ftest / **Wilcoxon**	GNB	NB	AODE	**GAODE**
NB	7-16-3			
	NO			
AODE	11-14-1	14-12-0		
	AODE	AODE		
GAODE	12-14-0	12-12-2	5-16-5	
	GAODE	GAODE	NO	
HAODE	13-13-0	13-12-1	6-19-1	6-18-2
	HAODE	HAODE	HAODE	HAODE

be thought to work better than GNB, but the number of datasets where GNB is not significantly worse than the best method, or actually is the best method, is 11, versus the 10 for NB. In fact, the Wilcoxon test returned no significant difference between these two methods for these datasets. We might expect the same reasoning to be extensible to the comparison between AODE and GAODE. However, this is not entirely true as the difference between means from the two algorithms is lower and the number of datasets where they are not significantly worse than the other classifiers is exactly the same. In this case, the Wilcoxon test also failed to show a significant difference.

Analysing these scores, we can confirm that both the GAODE and HAODE classifiers are significantly better than NB in any of its versions. As far as HAODE is concerned, not only does it obtain the highest accuracy mean, but also the highest number of datasets whose accuracies are not significantly different from the best one provided by any of the other classifiers. Likewise, according to the Wilcoxon test, it is significantly better than AODE and GAODE for this group of numerical datasets despite the considerable number of ties.

Furthermore, a Friedman test was performed for the 5 classifiers, yielding statistical difference. The posterior Nemenyi tests [Demšar, 2006; García & Herrera, 2009] only rejected the hypothesis that two algorithms are not significantly different in favour of GAODE and HAODE over GNB and NB, whereas AODE could not be proved to be significantly better than any of them.

4.4.2 Hybrid datasets

So far, we have seen the great capacity of HAODE as an alternative to AODE for numeric datasets. As opposed to GAODE, HAODE is able to deal with all kinds of datasets, hence we have also carried out experiments with 16 hybrid datasets included in a standard group of 36 UCI repository datasets, whose main characteristics are summarized in Table 4.4. All the numeric datasets in these group were included in the previous set of experiments and for the discrete datasets both classifiers are equal. That is the reason why in this block we just focus on hybrid ones.

Table 4.4: Main characteristics of the 16 hybrid datasets: number of attributes (n), number of classes (c), number of instances (m), number of discrete and continuous attributes ($\#D$ and $\#C$) and percentage of missing values ($\%M$).

Id.	Dataset	n	c	m	$\#D$	$\#C$	$\%M$
1	anneal.ORIG	38	6	898	32	6	63.32
2	anneal	38	6	898	32	6	0.00
3	autos	25	7	205	10	15	11.06
4	colic.ORIG	27	2	368	20	7	18.70
5	colic	22	2	368	15	7	22.77
6	credit-a	15	2	690	9	6	5.00
7	credit-g	20	2	1000	13	7	0.00
8	heart-c	13	2	303	7	6	0.17
9	heart-h	13	2	294	7	6	19.00
10	hepatitis	19	2	155	13	6	5.39
11	hypothyroid	29	4	3772	22	7	5.40
12	labor	16	2	57	8	8	0.00
13	lymph	18	4	148	15	3	0.00
14	sick	29	2	3772	22	7	5.40
15	vowel	13	11	990	3	10	0.00
16	zoo	17	7	101	16	1	0.00

Table 4.5 shows the accuracy results with NB (estimating Gaussians or multinomials according to the type of the attribute), AODE and HAODE using a 5x2cv on the evaluation and applying the discretization method previously mentioned in all the cases. The reason why the order of the datasets was altered will be given below.

Taking each w-t-l notation to mean that HAODE wins in w datasets, ties in t and loses in l datasets compared to AODE at a 95% confidence level, the hybrid classifier significantly improves on AODE in 1 of them, loses in 5 others and draws in 10 of them (1-10-5). Even though these are not the results we expected, considering only these hybrid datasets, it cannot be proved that there exists a significant advantage of AODE over HAODE, as Wilcoxon does not guarantee statistical difference.

Looking for a plausible explanation of this fact, specially taking into account the good results obtained by HAODE vs AODE in numerical datasets (Table 4.2), we analysed the percentage of numerical variables with respect to

Table 4.5: Accuracy results obtained with NB, AODE and HAODE classifiers in the hybrid datasets. The black square next to certain outputs means that the corresponding classifier on this particular dataset either obtains the highest accuracy or is not significantly worse than the classifier which does.

Id	NB	AODE	HAODE	$\%M$
16	■90.4950	■91.6832	■94.2574	0.00
13	■81.0811	■80.8108	■82.5676	0.00
15	50.6667	61.0505	■78.4444	0.00
7	■74.1600	■74.4400	■75.3200	0.00
12	■88.4211	■87.7193	■88.0702	0.00
2	■95.1448	■96.7483	■92.7840	0.00
8	■83.3003	■83.3003	■83.7624	0.17
6	■86.0290	■86.2609	78.8696	5.00
10	■82.3226	■83.0968	■84.3871	5.39
11	■97.7253	■98.0011	95.6416	5.40
14	97.0891	■97.2057	94.5652	5.40
3	■58.7317	■64.1951	■57.5610	11.06
4	■69.6196	■69.7826	60.8696	18.70
9	■83.8776	■83.9456	■83.4014	19.00
5	■79.3478	■81.0870	■78.8043	22.77
1	■93.1403	■93.9866	88.7751	63.32
Av	81.947	83.3321	82.3801	

discrete ones in hybrid datasets, but no significant pattern was found. Then, we turned to study the impact of missing values and, in this case, a relevant pattern can be obtained: the presence of missing values seems to punish HAODE vs AODE. Thus, in Table 4.5, hybrid datasets have been ordered according to their percentage of missing values. Above the line of the 2nd column in Table 4.5 we have the ones with almost no missing values. In fact, the Wilcoxon test shows statistical difference when only the datasets with missing values are considered. Based on the apparent tendency of HAODE to punish datasets with missing values, we then preprocessed all the datasets with an unsupervised filter to replace missing values with the modes and means from the existing data in the corresponding column. The same experiments were executed obtaining a result of 2-12-2. For the first group of numeric datasets the results are the same, as only breast-w has a $0,23\%$ of missing values. These results lead us to the conclusion that HAODE can be more sensitive to missing values than the other classifiers included in the comparison. It seems that the repeated use of different estimators (average of n models) made from few data when using Gaussian networks is more damaging than when they are made from multinomials.

4.5 Conclusions and future work

In this chapter, we have proposed two alternatives to AODE in order to deal with continuous attributes without performing a direct discretization process over the whole data. The first classifier, GAODE, applies CGNs to model the relationships between each predictive attribute and its parents, obtaining competitive results compared to AODE. GAODE implies a reduction in the space complexity and the parameters can be computed *a priori* in a single pass over the data, maintaining AODE's time complexity as well. This approach can also provide a more reliable and robust computation of the necessary statistics as the parameters are exclusively class-conditioned.

Furthermore, we have also presented a "hybrid" classifier, HAODE, which keeps the superparent attribute discrete in every model. This approach offers the clear advantage of dealing with any kind of dataset. Nonetheless, even though it is in general competitive when compared with AODE, it has shown a clear preference for datasets with continuous attributes and the absence of missing data, where it is significantly better than AODE.

Even though Gaussian networks often provide a reasonable approximation to many real-world distributions, they assume variables are sampled from Gaussian distributions. In the following chapter, we are exploring more general distribution probabilities, specifically the application of Mixtures of Truncated Exponentials (MTEs) [Moral *et al.*, 2001]) to AODE, which entails a more precise estimation, being also able to model Bayesian, Gaussian and hybrid networks.

Chapter 5

The MTE-AODE classifier

To be beyond any existing classification has always pleased me.
Boyd Rice. (1950-)
American artist

Abstract

As indicated in the previous chapter, AODE is exclusively defined to deal with discrete variables. Two approaches to avoid the use of the discretization pre-processing technique have been presented in Chapter 4, which involve, in lower or greater degree, the assumption of Gaussian distributions. In this chapter, we propose the use of mixtures of truncated exponentials, whose expressive power to accurately approximate the most commonly used distributions for hybrid networks has already been demonstrated. We perform experiments on the use of MTEs over a large group of datasets for the first time, and we analyse the importance of selecting a proper number of points when learning MTEs for NB and AODE, as we believe, it is decisive to provide accurate results.

5.1 Introduction

As most of the techniques based on Bayesian networks, AODE is defined to work with multinomial probability distributions, and hence, all continuous variables have to be treated somehow before the classification process. To this end, discretization techniques seem to be the most direct alternative, even though they entail inherent loss of information that may have a negative impact on the accuracy obtained when classifying.

In the previous chapter, two different proposals are described to deal with numerical variables in a different way [Flores *et al.*, 2009a]. Both proposals are based on the assumption that for each configuration of the categorical variables, all the numerical attributes are sampled from a Gaussian density distribution. Despite this strong assumption, Gaussian distributions usually provide reasonably good approximations to many real-world distributions.

In this sense, MTEs [Moral *et al.*, 2001] have become an attractive alternative, as they offer an exact frame for working with hybrid networks and the parameter estimation process from data, both for univariate and conditional potentials, is well-defined [Rumí *et al.*, 2006]. Therefore, in this chapter we generalize the use of the AODE classifier to work with all kind of datasets by estimating MTEs for all the density functions involved. Nevertheless, estimation using MTEs involves the selection of the maximum number of points into which the domain of numeric variables is partitioned. This parameter can be decisive to obtain accurate estimation from data, as shown below.

We will also study the application of MTEs over a large group of datasets without an individual parametrization, in order to find out if a general recommendation on how to best handle the continuous attributes can be given. So far, only studies tailored to specific datasets have been carried out using estimations based on MTEs.

We have performed the same study on NB as well, as it is directly extensible and always a good baseline to take into consideration.

The chapter is then organized as follows: Section 5.2 defines the proposed MTE-AODE classifier. In Section 5.3, we describe the experimental setup and results when comparing the performance of using MTEs with Gaussian distributions and discretization methods. Furthermore, we analyze the importance of selecting an adequate number of intervals when building the MTEs. Finally, Section 5.4 summarizes the main conclusions of this study and outlines the future work related with it.

5.2 MTE-AODE classifier

Definition 5.1 *(MTE-AODE classifier [Flores et al., 2011b]) Let $A_1, \dots, A_n$ be a set of (continuous and/or discrete) features and C a class variable. An MTE-AODE classifier is a model that classifies an individual described by features $(a_1, \dots, a_n)$ as belonging to the class c_{MAP} computed as in Equation 2.4 (MAP hypothesis for AODE), and where all the involved probability functions are of class MTE* [1].

Hence, this classifier keeps the original AODE's structure, as GAODE and HAODE.

The advantages of MTE-AODE over GAODE are that, in this case, we do not assume that the underlying distribution is of Gaussian type, as MTEs are able to accurately represent the most common distributions [Cobb *et al.*, 2006] and also, to handle datasets with discrete and continuous variables, i.e. hybrid datasets. With respect to HAODE, an added advantage is that there is no need to discretize the super-parent nodes, as the MTE paradigm can deal with discrete variables with continuous parents.

In turn, the drawback is that the learning phase is slower, as the parameters are adjusted iteratively following the algorithm described in Rumí *et al.* [2006]. Nevertheless, the speed of the learning phase can be tuned at the cost of the precision provided by the MTEs.

5.3 Experimental methodology and results

5.3.1 Decisions for the experimental frame

We have adopted two pre-processing steps to begin with. This is in order to make the group of datasets uniform and suitable for all the classifiers considered in the comparison:

- Unsupervised filter to replace all the missing values with the modes and means from the existing data in the corresponding column.

[1] Note that we can trivially extend this definition to the MTE-NB classifier as a particular case of MTE-AODE.

- Unsupervised filter to remove attributes that do not vary at all or whose variance percentage is greater than 99%.

Note that the number of intervals into which the domain for a continuous variable is split in an MTE potential is a parameter whose value remains to be tuned. Traditionally, it has been chosen based on empirical results, specifically, a trade-off between low complexity and high fitting power is desired. In this work, we have performed experiments partitioning the domain of each variable into 5 and 10 intervals (EF5 and EF10 respectively, from equal frequency division). Why these numbers? In Chapter 6, a comparison between different discretization methods for NB, AODE and HAODE (among other semi-naive BNCs) is carried out. In this study, the best results are obtained when applying EF discretization methods with 5 bins for AODE and 10 bins for NB and HAODE. Note that, in the MTE case, EF discretization is applied only to parent variables in a conditional distribution represented by a mixed tree. The domain of the variable that actually appears in the functional definition of the MTE potential in each leaf of the mixed tree is split taking into account the inflection and extreme points of the sample density [Rumí *et al.*, 2006], although a maximum number of intervals is set. Furthermore, we have made experiments with MTEs applying the supervised minimum-entropy-based discretization proposed by Fayyad & Irani (F&I) [Fayyad & Irani, 1993].

We have carried out experiments over the same group of 16 hybrid datasets considered in the previous chapter (Table 4.4, Section 4.4.2), using 5x2cv to perform the evaluation part. The software used in this case is Elvira [Elvira-Consortium, 2002], a tool in Java for probabilistic graphical models in continuous and discrete domains. The number of exponential terms in each of the MTEs estimated have been set to 2, which is the default value in Elvira.

5.3.2 Experimental results

Table 5.1 shows the accuracy results for this group of datasets when using Gaussians and EF5 for the following classifiers: NB using Gaussians (GNB);

NB, AODE and HAODE[1] applying EF5 (simply NB and AODE); and NB and AODE using MTEs with 5 intervals (MTE-NB and MTE-AODE). The bullet next to certain outputs indicates the best value when comparing the results provided by the three approaches based on NB on one hand (three first columns), and AODE on the other (three last columns). Similar results are shown for EF10 and F&I in Tables 5.2 and 5.3.

Table 5.1: Accuracy results obtained for NB, MTE-NB, AODE and MTE-AODE in the hybrid datasets (EF5).

	Naive Bayes			AODE		
Id	GNB	NB	MTE-NB	AODE	HAODE	MTE-AODE
1	66.8597	•88.5078	82.5612	•88.4633	81.9376	82.2717
2	87.1938	•92.9399	90.2227	•96.8597	94.4098	93.2294
3	56.6829	•60.6829	59.2109	68.4878	•70.9268	62.0350
4	71.5761	•78.2609	68.6957	•80.3261	73.0978	69.5652
5	77.5000	74.5652	•80.4891	74.4565	75.7065	•81.7935
6	77.5362	•86.1159	84.4348	•86.4638	80.7826	85.3333
7	•74.7600	74.6200	74.5600	74.1600	73.7400	•74.7400
8	82.6403	•84.0924	82.6965	•83.2343	82.4422	82.5000
9	81.9048	83.4694	•84.1497	83.6735	81.7687	•84.2177
10	83.2258	83.4839	•85.0316	82.1935	83.2258	•84.2524
11	•95.4189	94.5758	94.1569	94.6235	•95.3340	93.8388
12	91.2281	•93.6842	90.1847	•93.3333	90.8772	91.2562
13	81.7568	82.7027	•83.2432	83.3784	82.2973	•83.7838
14	92.3065	92.1898	•94.3584	93.3033	•95.5885	94.1676
15	•59.5152	53.6970	55.1313	75.5960	•86.1010	70.7879
16	92.6733	90.6931	•93.6627	91.4851	•94.4554	93.8588
Av	79.5487	82.1426	81.4244	84.3774	83.9182	82.9770

All pairwise comparison between every algorithm with MTE-NB and MTE-AODE are summarized in Table 5.4, where each entry w-l in row i and column j means that MTE-NB or MTE-AODE in row i wins in w datasets and loses in l datasets, compared to either G (Gaussian NB), D (NB or AODE applying the type of discretization indicated in row i) or H (HAODE), depending on the content in column j.

In the case of NB, the use of MTEs offers an overall improvement over the use of Gaussians. However, the conclusions are not as clear to obtain when comparing with the use of discretization, where the latter specially stands out when using 10 bins. The difference is less clear when 5 bins or F&I

[1]Note that the GAODE can not be applied in this context as it is restricted to work exclusively with numeric attributes.

Table 5.2: Accuracy results obtained for NB, MTE-NB, AODE and MTE-AODE in the hybrid datasets (EF10).

		Naive Bayes			AODE	
Id	GNB	NB	MTE-NB	AODE	HAODE	MTE-AODE
1	66.8597	•91.1804	82.6726	•86.0356	82.6281	82.0935
2	87.1938	•93.5189	90.3341	•96.1470	94.4321	92.9621
3	56.6829	•66.9268	59.8934	64.0824	•70.1463	63.0164
4	71.5761	•78.2065	69.1304	73.0978	•73.5870	69.4565
5	77.5000	74.5652	•80.7609	81.3587	75.0000	•81.6848
6	77.5362	•85.7971	84.2029	84.3188	80.9855	•84.9275
7	•74.7600	74.6000	74.4200	74.9200	74.1400	•75.1000
8	82.6403	•82.8383	82.5645	81.9715	82.1122	•82.5658
9	81.9048	82.8571	•84.0136	•84.0136	80.5442	83.6054
10	83.2258	83.4839	•85.0316	•85.0266	83.4839	84.2541
11	•95.4189	95.3446	94.1729	94.4274	•95.4401	93.7964
12	91.2281	•92.2807	90.1847	90.8867	90.5263	•91.2562
13	81.7568	82.5676	•83.2432	•84.1892	82.4324	83.7838
14	92.3065	•95.1326	94.3054	•96.2725	96.0233	94.5387
15	•59.5152	57.8788	55.5758	73.8990	•88.6263	71.3333
16	92.6733	90.6931	•93.6627	94.0549	•94.4554	93.8588

Table 5.3: Accuracy results obtained for NB, MTE-NB, AODE and MTE-AODE in the hybrid datasets (F&I).

		Naive Bayes			AODE	
Id	GNB	NB	MTE-NB	AODE	HAODE	MTE-AODE
1	66.8597	•89.7327	81.2695	•89.8441	78.7528	81.4477
2	87.1938	•94.4543	89.6659	•96.8597	92.7840	92.9621
3	56.6829	58.8293	•59.6031	63.9024	•68.6829	63.5028
4	71.5761	•72.8804	69.7826	74.0761	•74.5652	70.7065
5	77.5000	79.9457	•80.1087	81.2500	77.8261	•81.6304
6	77.5362	•86.0290	83.7101	•86.2319	79.0725	84.8986
7	•74.7600	74.1600	74.4400	74.4400	•75.3200	75.0800
8	82.6403	•83.2343	81.4434	•83.2343	82.9043	83.0281
9	81.9048	83.6735	•83.8776	83.6735	83.1293	•84.2177
10	83.2258	82.3226	•83.4898	83.2258	82.5806	•84.9018
11	95.4189	•97.9586	94.2312	•98.2397	95.5567	93.8600
12	•91.2281	87.3684	88.4236	88.4211	•92.9825	90.1847
13	81.7568	81.0811	•83.3784	80.8108	82.5676	•83.6486
14	92.3065	•97.0042	94.2948	•97.1686	93.9449	93.6638
15	•59.5152	50.6667	51.7980	61.0505	•78.4444	69.5152
16	92.6733	90.6931	•93.6627	91.4851	•94.2574	93.8588

are applied. Similar reasoning is extended to AODE when comparing MTEs and the use of discretization, where even significant improvement is obtained when the Wilcoxon test is performed in EF10. As far as the comparison between HAODE and MTEs is concerned, we find also very competitive results. Note that here, we should focus on the domain of the variable that appears in each leaf of the mixed tree, as the partition of the domain for the parent variables in a conditional distribution represented by a mixed tree is directly determined by the discretization method selected.

Table 5.4: Accuracy comparison between NB and AODE with MTEs, and other approaches to deal with continuous variables. The black triangle indicates significant performance applying the Wilcoxon test.

	Naive Bayes		AODE	
	MTE vs G	MTE vs D	MTE vs H	MTE vs D
EF5	11-5	7-9	9-7	7-9
EF10	10-6	5-11	8-8	▼5-11
F&I	10-6	9-7	8-8	8-8

5.3.3 Discussion

In the light of the results, we observe that the selection of the proper way to deal with continuous variables in NB and AODE depends in high degree on the dataset. This is the reason why in this section, we are analysing the importance on the selection of the number of cutpoints when estimating the MTEs.

Table 5.5 shows the accuracy results for MTE-NB and MTE-AODE when using 5EF, 10EF and F&I to indicate the number of intervals into which the domain of each leaf node will be partitioned to estimate the MTEs, and also, in the case of AODE, to create the intervals for the superparents.

We can observe how the results can dramatically change depending on the number of intervals selected. There are some few datasets where accuracy remains the same, usually due to the fact that the number of intervals created is lower than specified (e.g. for attributes of type integer).

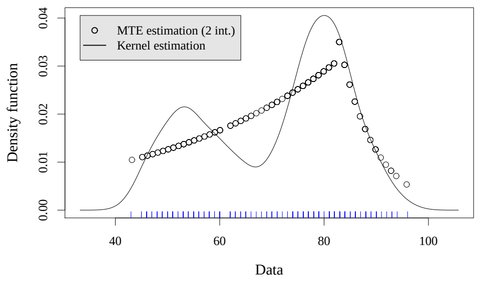
Figures 5.1 (a), (b) and (c) show the kernel and MTE's densities estimated for a numeric attribute called `waiting`, which represents the waiting time between eruptions for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA [Azzalini & Bowman, 1990]. Different values to set the maximum number of intervals have been used to estimate the MTEs and

Table 5.5: Accuracy results obtained for MTE-NB and MTE-AODE when using 5EF, 10EF and F&I to create the intervals.

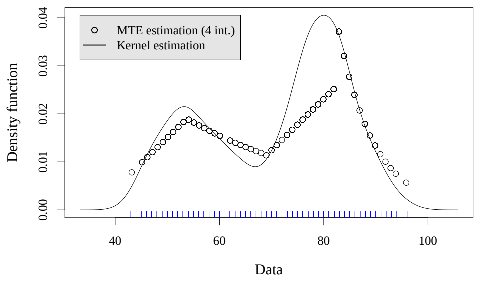
	Naive Bayes			AODE		
Id	5EF	10EF	F&I	5EF	10EF	F&I
1	82.5612	•82.6726	81.2695	•82.2717	82.0935	81.4477
2	90.2227	•90.3341	89.6659	•93.2294	92.9621	92.9621
3	59.2109	•59.8934	59.6031	62.0350	63.0164	•63.5028
4	68.6957	69.1304	•69.7826	69.5652	69.4565	•70.7065
5	80.4891	•80.7609	80.1087	•81.7935	81.6848	81.6304
6	•84.4348	84.2029	83.7101	•85.3333	84.9275	84.8986
7	•74.5600	74.4200	74.4400	74.7400	•75.1000	75.0800
8	•82.6965	82.5645	81.4434	82.5000	82.5658	•83.0281
9	•84.1497	84.0136	83.8776	•84.2177	83.6054	•84.2177
10	•85.0316	•85.0316	83.4898	84.2524	84.2541	•84.9018
11	94.1569	94.1729	•94.2312	93.8388	93.7964	•93.8600
12	•90.1847	•90.1847	88.4236	•91.2562	•91.2562	90.1847
13	83.2432	83.2432	•83.3784	•83.7838	•83.7838	83.6486
14	•94.3584	94.3054	94.2948	94.1676	•94.5387	93.6638
15	55.1313	•55.5758	51.7980	70.7879	•71.3333	69.5152
16	93.6627	93.6627	93.6627	93.8588	93.8588	93.8588
Av	81.4244	81.5105	80.8237	82.9770	83.0146	82.9442

the resulting densities have been plotted and compared to the kernel density. Figure 5.1 (a) shows how selecting a number of intervals too small can incur a too generalized estimation of the original data. This, in terms of discretization bias and discretization variance [Yang & Webb, 2009] can imply a renunciation of an improvement in terms of bias while maintaining quite a low variance, as estimations are made from large amount of samples. On the other hand, if the maximum number of intervals is too high, the obtained effect is the opposite: bias might get lower at the expense of an increase in terms of variance and loss of generalization capabilities. Hence, the smaller the number of intervals selected, the lower also the expressive power of the MTEs; in turn, the larger is this number, the higher also will be the risk of overfitting the training data. In fact, additional tests have been performed by setting no restrictions in the maximum number of intervals, i.e., just taking into account the inflection and extreme points of the sample density, and the results improved 11-0-5 as for using F&I for example.

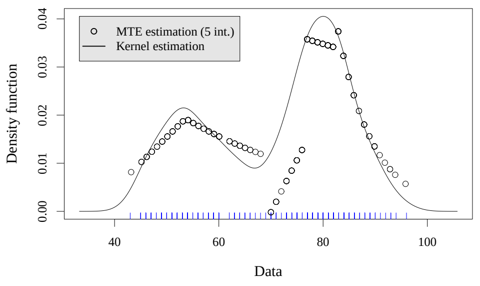
That is why, we believe that the selection of the number of intervals in this case is very important. It is not only advisable to find the optimum for every dataset, but also, to find the optimum for every domain to partition



(a) 2 intervals



(b) 4 intervals



(c) 5 intervals

Figure 5.1: Estimation of MTEs when selecting different number of cutpoints.

in each leaf of the mixed tree independently, requiring a more sophisticated supervised "discretization" technique oriented to the estimation of MTEs.

As for the time complexity is concerned, the estimation of MTEs requires more than performing EF discretization. However, this time can be controlled in different ways: on one hand, limiting the number of exponential terms in each of the estimated MTEs, but also, playing with the maximum number of intervals into which divide the domain. Hence, being able to deal with very large datasets at a lower cost.

5.4 Conclusions and future work

In this chapter, we have proposed an alternative approach to generalize the application of AODE to datasets with continuous and/or discrete attributes. To this end, we have resorted to the use of MTEs, specifically, all the probability functions now involved in the AODE classifier are of class MTE; as this kind of distributions is able to represent the conditional probability functions in a Bayesian network without any structural restriction, providing, at least in theory, a high expressive power. As we have shown, this idea is directly extensible to NB (where the use of Gaussians is trivial, unlike in AODE); and it is also to other semi-naive Bayesian classifiers.

So far, all the approaches designed to avoid direct discretization of the numeric attributes in AODE assumed, in lower or greater degree the existence of Gaussian distributions. Although this is a reasonable approximation for some problems, it does not hold in many other datasets, as we have corroborated in this work, and that is why the application of MTEs is an option to consider.

Nevertheless, the use of MTE estimations requires selecting the proper number of intervals into which the domain of leaf variables in the mixed tree is split, in order to compete with discretization methods. Also, we have seen how AODE is less sensitive to the number of cutpoints selected compared to NB. We believe this is so, due to its own definition as an aggregation of models.

Finally, we propose the study of a new supervised method to dynamically search for the optimum number in every case into every dataset. The idea is to find a good trade-off between fitting and generalization capability of the model.

Part III

Discretization techniques for semi-naive BNCs

Chapter 6

Disjoint discretization techniques

In theory, theory and practice are the same. In practice, they are not.
Albert Einstein. (1879-1955)
German theoretical physicist

Abstract

Despite the loss of information discretization entails, it is a direct easy-to-use mechanism that can offer some benefits and, even though there are many ways to deal with continuous variables other than discretization (see Chapters 4 and 5), it is still commonly used. This chapter presents a study of the impact of using different discretization strategies on a set of representative BN classifiers. With this comparison we analyse to what extent the type of discretization method affects classifier performance in terms of accuracy and bias-variance discretization. Our main conclusion is that, even if a discretization method produces different results for a particular dataset, it does not really have an effect when classifiers are being compared. That is, given a set of datasets, accuracy values might vary, but the classifier ranking is generally maintained. This is a very useful outcome, as assuming that the type of discretization applied is not decisive, future experiments can be d times faster, d being the number of discretization methods considered.

6.1 Introduction

Discretization is one of the pre-processing techniques most broadly used in machine learning and data mining. Strictly speaking, by means of a discretization process the real distribution of the data is replaced with a mixture of uniform distributions. In practice, discretization can be viewed as a method for reducing data dimensionality, since the input data are transformed from a huge spectrum of numeric values to a much smaller subset of discrete values, normally by placing variable values into ranges.

In Section 2.2.1, only a few of the many different discretization techniques are shown. There is no doubt that, when dealing with a concrete problem (dataset), choosing a certain discretization method can have a direct impact on the success (accuracy, AUC, etc.) of the posterior classification task. However, in this chapter we do not focus on the effect of the chosen discretization method on a specific application domain (dataset), but its impact when studying a BN classifier over a significant range of application domains (datasets). That is to say, *should we worry about the discretization method applied when designing the set of experiments in order to study whether or not the analysed method is better than other BN classifiers?* If the answer is yes, then we must add a new parameter (the discretization method) to our experimental study, and so the number of experiments will be multiplied by d, d being the number of tested discretizations. Otherwise, if the answer is no, then we can avoid introducing this parameter in the experimental study and therefore we will save a considerable amount of time in our experiments (to be precise our experiments will be d times faster).

In order to answer the question posed above, in this study, we intend to perform an empirical analysis of this problem, taking as our basis a subset of classifiers based on BNs: NB, TAN [Friedman *et al.*, 1997], KDB Classifier [Sahami, 1996], AODE [Webb *et al.*, 2005], HAODE [Flores *et al.*, 2009a] and a more general BN classifier which uses a Hill Climbing algorithm (BNHC) [Buntine, 1996]. We have seen in Chapter 4, how HAODE provides better results than AODE for continuous attributes when applying Fayyad and Irani's discretization method, but does it hold for other kinds of discretization techniques? [Flores *et al.*, 2010, 2011a].

With respect to the datasets considered in our experiments, we have used a significant sample consisting of the 26 datasets from the UCI archive [Frank & Asuncion, 2010] presented in Chapter 4 (Table 4.1). Although more classifiers (and perhaps datasets) can be added to our test suite in the future, we think that the current study is already a significant one to draw the first conclusions.

The rest of the chapter is organized as follows: Section 6.2 presents both the design of the experiments performed and their results. It is comprised of the experimental frame and three sets of experiments: global analysis, k value selection and the study of NB-tailored discretization techniques. Finally, Section 6.3 summarizes the main conclusions of this study.

6.2 Experimental methodology and results

6.2.1 Experimental frame

In the following three subsections, the three groups of experiments carried out are detailed: Subsection 6.2.2 includes an extensive study, in terms of accuracy and error components (bias and variance), of the performance obtained by the BN classifiers when using six discretization methods; Subsection 6.2.3 explains why the maximum number of parents for a node in the KDB algorithm is set to 1, as we will see below; and finally, Subsection 6.2.4 extends the comparison to two other NB-tailored discretization techniques with the aim of finding out if the behaviour observed in Subsection 6.2.2 can be extended to this kind of discretization techniques as well.

6.2.2 Experiment 1: global analysis and results

The experiments are performed over the following six BNCs: NB, TAN, KDB1 (KDB with $k = 1$), BNHC, AODE and HAODE. We consider 6 different techniques for discretizing the datasets: equal-width discretization with 5 (EW5) and 10 bins (EW10) and optimizing the number of bins through entropy minimization (EWE); equal frequency discretization with 5 (EF5) and 10 bins (EF10); and Fayyad and Irani's supervised discretization method (F&I). In all cases, the corresponding filters included in WEKA [Hall *et al.*, 2009] for these types of discretizations are applied.

For all the experiments 5x2cv is used. The bias-variance decomposition is performed using the sub-sampled cross-validation procedure exactly as specified in Webb & Conilione [2002].

We analyse the results obtained in terms of accuracy and error on different blocks, the latter divided into the bias and variance components according to Webb [2000].

In order to provide a descriptive comparison between the results provided by the different classifiers considering the various types of discretization methods, we show the average measures in terms of accuracy (Subsection 6.2.2.1) and error rate divided into bias and variance (Subsection 6.2.2.2). This information is extremely useful as it provides a very compact and visual representation of the comparative. However, as we are dealing with different domains its results may not be commensurable and that is why some statistical tests are carried out to provide a more analytical point of view.

In this sense, and following Demšar [2006] and García & Herrera [2009] guidelines, we have decided to use Friedman tests in two levels of abstraction: the first one corresponds to compare the performance of the different discretization methods for each classifier (here, if the discretization method does not matter, no differences should be found), and the second, to compare the different classifiers for a specific discretization method (here, if the discretization method does not matter, the same differences should be found among the classifiers in every discretization method).

The Friedman test is a non-parametric statistical test similar to ANOVA (Fisher [1959]) that additionally ranks the algorithms compared. Iman & Davenport [1980] came out with a less conservative statistic than Friedman's, that we are applying as well. In order to compare all the classifiers with each others we use the well-known Nemenyi test (Nemenyi [1963]), which is similar to the Tukey test for ANOVA. Note that, in this case, it is not of major interest to compare the performance of a particular classifier (con-

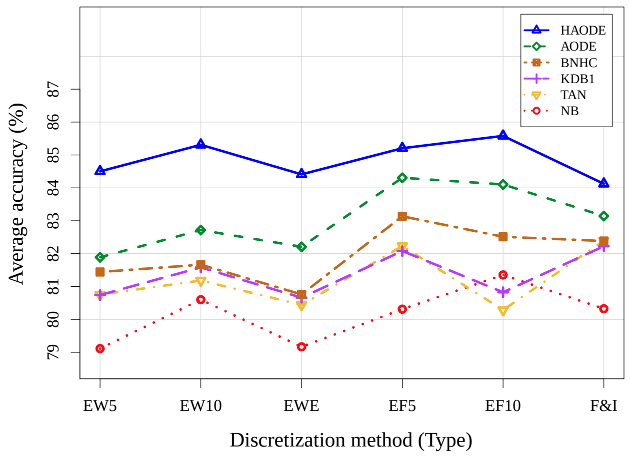


Figure 6.1: Comparison of average accuracy for NB, TAN, KDB1, AODE and HAODE when using different discretization methods.

trol classifier) with the rest and that is why other tests, such as Holm's or Bonferroni-Dunn's, have not been used here.

6.2.2.1 Study in terms of accuracy

The Y-axis in Figure 6.1 represents the average accuracy for the 26 datasets considered. The different lines correspond to the behaviour of the six classifiers for the 6 discretization methods tested, which are represented on the X-axis. Although the tendency followed by the six classifiers is similar, and more importantly, the ranking among classifiers is maintained in all cases except for NB with TAN and KDB1 when discretizing with EF10, we can see that equal frequency discretization is specially good for AODE, whereas F&I performs worse for HAODE. We can also observe how HAODE, from among the 6 classifiers tested, is the least sensitive to the discretization method applied (its corresponding line looks smoother than the rest).

As indicated above, in this comparison we have also included a more generic Bayesian classifier named BNHC to construct an augmented Bayesian network (ABN), from Bayes net with hill climber, as in WEKA[1]. This clas-

[1]This Bayes network learning classifier uses a hill-climbing algorithm for adding, delet-

sifier has been introduced for the sake of curiosity, as it does not belong to the family of semi-naive BNCs. But despite the fact that it performs a more exhaustive search of the structure of the network, it provides an average accuracy between that of KDB and AODE. Furthermore, its training time is around 453 times worse than KDB and 642 times worse than AODE (for the dataset `mfeat-factors`, for example).

All pairwise comparisons between every classifier with each other are summarized in Table 6.1, where each entry w in row i and column j means that HAODE (the top classifier) wins in w datasets, compared to the algorithm in column j, and using the discretization method in row i. The idea here is that all the w values in the same column should be as similar as possible, meaning that regardless of the discretization method applied, the top classifier wins approximately in the same number of datasets. Hence, the biggest difference we found is only 4 datasets (out of 26) when HAODE is compared to BNHC and AODE.

Table 6.1: Pairwise comparisons between HAODE and the rest of classifiers.

	HAODE				
	NB	TAN	KDB1	BNHC	AODE
EW5	20	21	21	18	23
EW10	20	21	23	20	21
EWE	23	24	24	22	22
EF5	20	22	23	20	21
EF10	21	22	23	21	21
F&I	22	22	22	22	19
Max. difference	3	3	3	**4**	**4**

Figure 6.2 shows, on each circular graph, the individual accuracy obtained over each dataset for the six classifiers. Each graph corresponds to a specific discretization method, from top to bottom and left to right: EW5, EW10, EWE, EF5, EF10 and F&I. The circumferences are divided into 26 sectors (every $\approx 13.85°$) corresponding to the different datasets. The radius, in turn,

ing and reversing arcs. The maximum number of parents a node in the Bayes net can have is set to 5. The initial network used for structure learning is a NB network. This hill climber also considers arrows as part of the NB structure for deletion.

represent the percentage of accuracy for each dataset (from 50 to 100). We can see, from the circular visualization, that HAODE almost always encloses the other methods, indicating that it dominates them in terms of accuracy. Similarly, the line corresponding to AODE covers TAN's, KDB1's and NB's; KDB1's and TAN's almost overlap and they enclose NB's in most of the cases. However, the situation is different for BNHC, where the pattern is, in general, more irregular. It makes sense though, as it is the only one which does not belong to the semi-naive family of BNCs.

Comparisons between discretization methods: As indicated above, Friedman tests were applied to perform the multiple comparison of the different discretization methods for each classifier, as well as the Nemenyi post-hoc test, using the software provided by the guidelines in García & Herrera [2009]. The summarized results are shown in Table 6.2.

Table 6.2: Test results when comparing the discretization methods over each classifier (in brackets, the p-value obtained). The null hypothesis (H_0) states that there is no difference between the algorithms. $\alpha = 0.05$ for all the cases.

	FRIEDMAN	**IMAN-DAV.**	**NEMENYI**
NB	Reject H_0 (0.034)	Not necessary	• None
TAN	Reject H_0 (0.006)	Not necessary	• F&I vs (EWE&EF10) (0.007 & 0.012)
KDB1	Reject H_0 (0.029)	Not necessary	• F&I vs EF10 (0.022)
BNHC	Accept H_0 (0.294)	Accept H_0 (0.296)	• None
AODE	Accept H_0 (0.069)	Accept H_0 (0.065)	• None
HAODE	Accept H_0 (0.052)	Reject H_0 (0.049)	• None

For NB, even though Friedman's claims statistical difference, these differences are not found by the post-hoc tests. As far as TAN is concerned, Nemenyi finds differences between F&I vs EWE and EF10; similar results are obtained for KDB1 except for the EWE difference. BNHC, AODE and

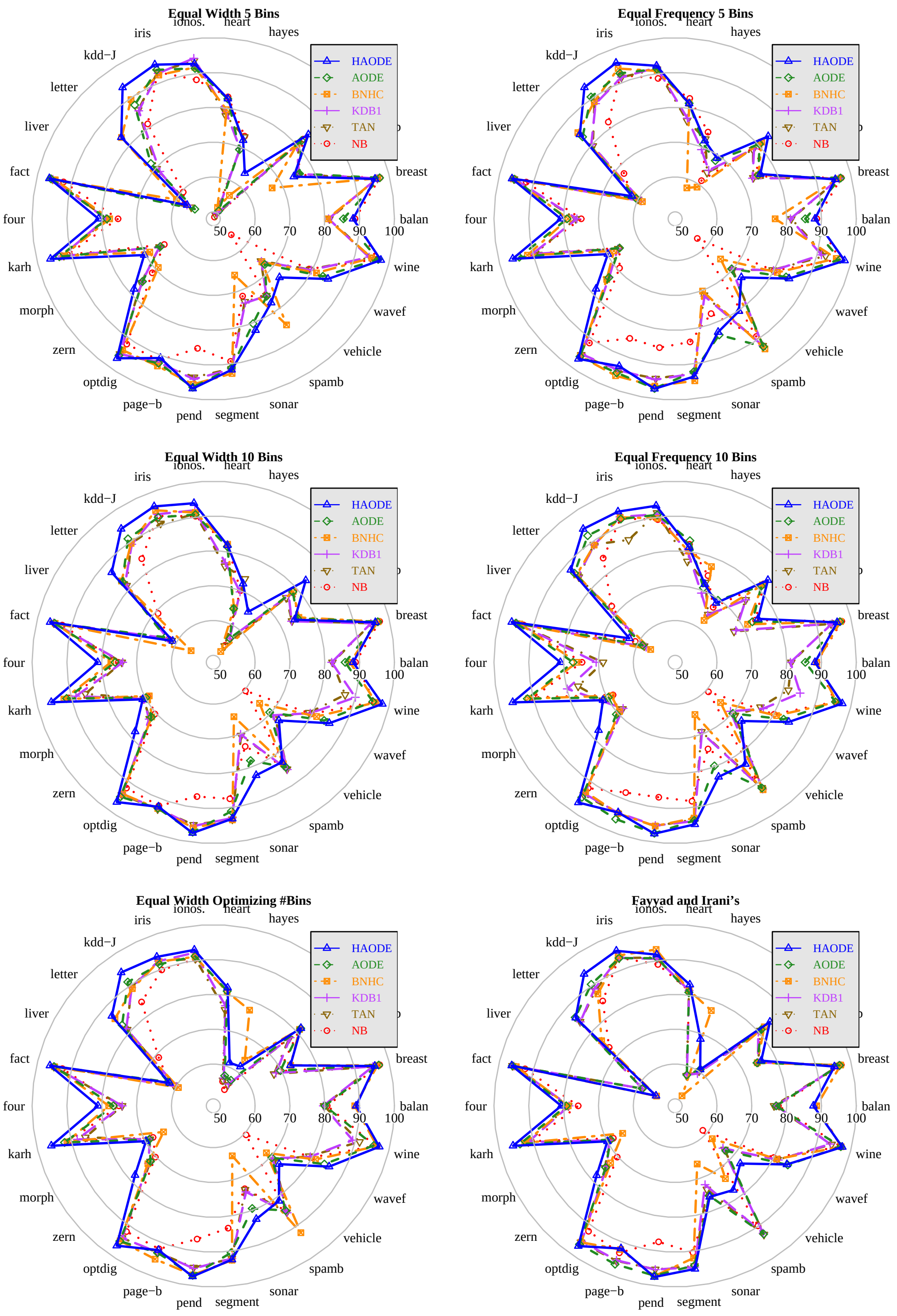


Figure 6.2: Comparison of accuracy obtained on each of the 26 datasets with NB, TAN, KDB1, BNHC, AODE and HAODE for the different discretization methods. From top to bottom and left to right: EW5, EW10, EWE, EF5, EF10 and F&I. The circumferences are divided into 26 sectors corresponding to the different datasets. The radius, in turn, represent the percentage of accuracy for each dataset.

HAODE seem to be the most robust to the discretization method, as the null hypothesis is rejected by Friedman.

Note that in this case, the Bonferroni correction has not been applied, as it is considered to be too conservative. Nevertheless, if the Bonferroni correction were to be applied, not a single difference would be found[1].

Comparisons between classifiers: If we change the point of view and perform the same tests comparing the different classifiers for a specific discretization method, we obtain evidence of statistical difference in all cases when applying the Friedman test. According to Nemenyi tests, HAODE is significantly better than NB, TAN, KDB1 and BNHC in all cases (except for BNHC in EW5). This test also states that AODE is better than NB when EW5, EF5 and F&I are used and KDB for EF5 and EF10. One interesting observation is that, according to Friedman, EWE is the discretization method that has found the clearest difference, as opposed to EW10. EW10 finds the smallest number of differences with Nemenyi also (3 rejections), as opposed to EF5 and EF10 (with 6 rejections).

It is worth noting that in all cases HAODE is placed in first position, AODE in second and BNHC in third by the ranking performed by the Friedman test. NB gets fourth position when EF10 is used, forcing both KDB1 and TAN to occupy fifth position in this case.

6.2.2.2 Study in terms of bias and variance

In Yang & Webb [2009] the behaviour of NB in terms of bias and variance is studied, and the authors refer to *discretization bias and variance*. We have endeavoured to carry out a similar analysis here, extending it to the rest of the BN classifiers considered in this chapter.

The error component can be divided into three terms: bias, variance and an irreducible term [Webb, 2000]. The **bias** describes the part of the error component that results from the systematic error when learning the algorithm, whereas the **variance** describes the random variation existing in the

[1] As the number of comparisons is equal to 15, the statistical significance level would then be $0.00\bar{3}$ instead of 0.05.

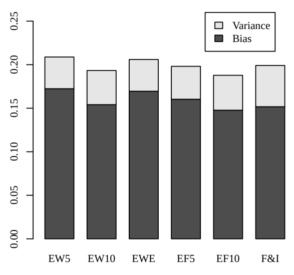
training data and from the random behaviour when learning the algorithm. The more sensitive the algorithm is, the higher the variance becomes. The irreducible error describes the error existing in an optimal algorithm (noise level in data).

There are two important concepts to be taken into account when trying to reduce the error components mentioned above: the number of intervals and the number of training instances contained in each interval. Intuitively, discretization resulting in large interval numbers tends to have low bias (any given interval is less likely to include a decision boundary of the original numeric attribute), whereas discretization resulting in intervals with a large number of instances tends to have low variance (as the probability estimations are more stable and reliable). The problem is that supposing there is a fixed dataset size, the larger the number of intervals, the smaller the number of instances per interval is.

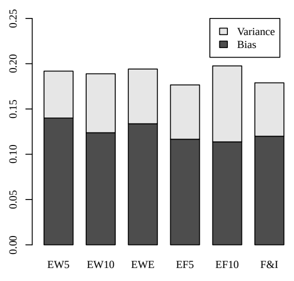
In Figure 6.3 the average error for every discretization method over all the datasets, divided into bias and variance, is shown. Now again HAODE obtains, on average, the lowest error rates. Similar reasoning is obtained when just the bias is considered, followed by AODE, KDB1 and TAN, which obtain analogous values. In terms of variance the analysis is slightly different. It is now NB that obtains the lowest variance in almost all cases, whereas TAN, KDB and BNHC, obtain the highest rates for all the discretization methods. Note that TAN's and KDB's results are similar in terms of bias to AODE's, but it is when considering the variance of the former that the final error is dramatically greater.

6.2.3 Experiment 2: justification for the parameter k being equal to 1 in KDB

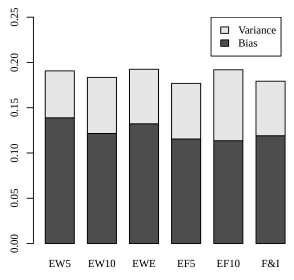
As mentioned above, KDB is a more flexible classifier compared with TAN, as it does not restrict the number of parents allowed for a feature to one, in addition to the class variable. However, this means that in practice, the maximum number of parents of a variable (k) must be fixed beforehand. As far as we know, a way to automatically identify an optimum k-value for a



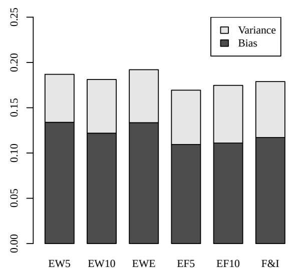
(a) NB



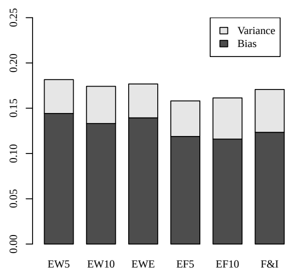
(b) TAN



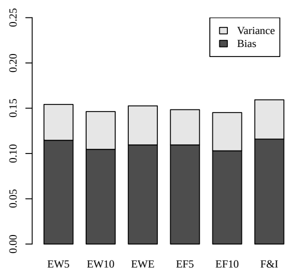
(c) KDB



(d) BNHC



(e) AODE



(f) HAODE

Figure 6.3: Mean error divided into bias and variance for NB, TAN, KDB, BNHC, AODE and HAODE. The discretization methods are presented on the X-axis, the Y-axis indicates the error rate.

given problem has not been determined yet[1]. Hence, for this study, in a similar way to Sahami [1996], we try with values of k equal to 1, 2 and 3

[1]Although it has been demonstrated that a good selection of the k-value, individually for each variable, provides significant improvements in terms of accuracy [Rubio & Gámez, 2011]

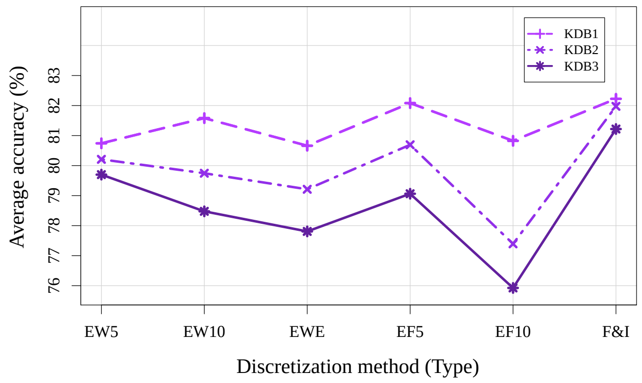


Figure 6.4: Comparison of average accuracy for KDB with different k values when using different discretization methods.

($k = 0$ being equivalent to NB). The average accuracy results are shown in Figure 6.4.

As KDB with $k = 1$ (KDB1) obtains better results in terms of average accuracy compared with the others, we have only included this one in the previous and following comparisons.

The reader might find this KDB1 too similar to the TAN classifier. In fact, from the point of view of the final structure created by both methods, they are. However, there are some differences in the way they have been built. To start off, TAN uses a Bayesian score metric, as implemented in WEKA[1] to evaluate the candidate structures, whereas the KDB1 method is implemented as specified in Sahami [1996], and hence, the way the structure is gradually formed by the two methods is different.

[1]Specifically, the Bayesian metric applied can be defined as in Bouckaert [2005]:

$$Q_{Bayes}(D) = \prod_{i=0}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})},$$

where D is the dataset, q_i the cardinality of the parent set of A_i in the network structure, r_i $(1 \leq i \leq n)$ the cardinality of A_i, N_{ij} $(1 \leq i \leq n, 1 \leq j \leq q_i)$ denotes the number of records in D for which $pa(A_i)$ takes its jth value and N_{ijk} $(1 \leq i \leq n, 1 \leq j \leq q_i, 1 \leq k \leq r_i)$ the number of records in D for which $pa(A_i)$ takes its jth value and for which A_i takes its kth value. $\Gamma(.)$ is the gamma-function, N'_{ij} and N'_{ijk} represents choices of priors on counts restricted by $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$.

6.2.4 Experiment 3: NB-tailored discretization techniques extended to other BN classifiers

In this section we study the application of two discretization techniques tailored to the NB classifier [Yang & Webb, 2009] to the other classifiers considered so far: TAN, KDB1, AODE and HAODE[1].

These two techniques, known as *proportional discretization* and *fixed frequency discretization*, are explained below. The aim of these discretization methods is to reduce both the bias and the variance obtained by NB when applied to the discretized data.

1. **Proportional discretization (PD)**: The idea behind this discretization method is to equally weight bias and variance reduction by setting both the number of the intervals and the number of instances per interval in such a way, that they are equally proportional to the size of the dataset. If we consider to apply PD to a numeric attribute for which there are m training instances with known values, the number of intervals (or bins) b and the number of values per interval s, is set considering that $b \cdot s = m$ and $b = s$. Hence, the method is equivalent to EFD with $b = \sqrt{m}$ bins.

   The advantage of this method is that any increase in training data would reduce both discretization bias and variance, as b and s increase. So, in theory, PD is bound to provide good results for large datasets (at least for NB [Yang & Webb, 2001]).

2. **Fixed frequency discretization (FFD)**: This technique pursues the same objective of keeping bias and variance under control, specially the latter. This is carried out by setting a minimum number of values, r, per interval. As the optimal value for r may vary from domain to domain, it is proposed to choose a value $r = 30$, since it is commonly accepted as the minimum sample size from which one should draw

[1]BNHC, however, has been left out as the high number of values (intervals) per attribute created by PD and FFD, makes it infeasible for BNHC to be executed in terms of RAM memory.

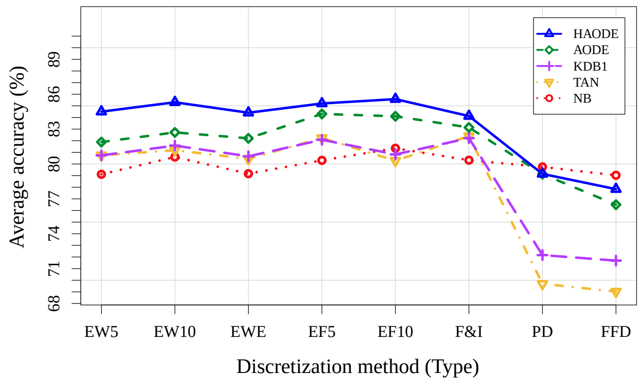


Figure 6.5: Comparison of average accuracy for NB, TAN, KDB1, AODE and HAODE when using different discretization methods (including PD and FFD).

statistical inferences [Weiss, 2002]. As the number of intervals is not limited a priori, more bins can be formed as the training data increase, and hence, the bias component is kept under control also.

Figure 6.5 shows the same accuracy results as Figure 6.1, but including the results when PD and FFD are used to discretize the datasets.

It seems not surprising that the only classifier whose averages are maintained is NB, as this kind of discretization is “designed” to suit this classifier [Yang & Webb, 2001]. However, even for NB, although both kinds of discretization are aimed to keep both bias and variance under control, in our experiments it is only accomplished for the first component (see Figure 6.6). The reason might be that the high dimensions of some of the datasets implies a number of intervals per attribute which is too high, with a relatively low frequency of instances per attribute. This would also partially explain why the more complex classifiers would perform dramatically worse with this kind of discretization methods, as their estimates are even less strong consistent estimates according to the strong law of large numbers [Casella & Berger, 2001; John & Langley, 1995].

It is then desired to perform a wiser discretization controlling both bias and variance for the rest of the classifiers that are different from NB. To this

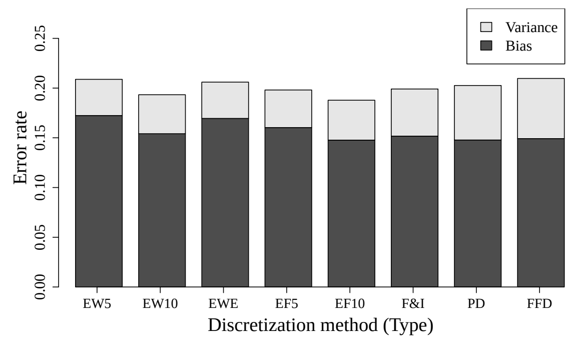


Figure 6.6: Mean error divided into bias and variance for NB. The discretization methods are presented on the X-axis (including PD and FFD), the Y-axis indicates the error rate.

aim, a discretization method that establishes different decision boundaries for an attribute depending on the test instance might be required, as it would depend on the specific values of the other attributes in that instance. It would also imply extending the notion of decision boundaries to the case of multiple attributes, which is beyond the scope of this chapter. For further work related to this, please refer to Hsu *et al.* [2000, 2003]; Yang & Webb [2009].

6.3 Conclusions and future work

In this chapter we have studied the effect, in terms of accuracy, bias and variance, of applying some of the most common discretization methods to NB, TAN, KDB, AODE and HAODE.

One of our first goals was the comparison between AODE and HAODE, in order to study if the results in Chapter 4 can be extensible to other discretization methods. The results obtained reveal that in all cases HAODE's average accuracy is higher than AODE's, the former being significantly better than the latter in all cases except when EF10 is applied.

Furthermore, it was of major interest to investigate whether the application of a particular discretization technique could alter the ranking of classifiers according to the accuracy obtained for the six BN classifiers taken into consideration. The results indicate that no matter what discretization method we use the ranking is the same for HAODE, AODE and BNHC with the rest of the classifiers, as their performance is sufficiently different. However, as NB, TAN and KDB obtain very similar results, their position in the ranking can vary in a particular case. Even so, in the light of the results, we believe that if the set of datasets is large enough, the discretization method applied becomes irrelevant when comparing the BN classifiers.

Nonetheless, we have also seen that, in the case where the discretization technique is tailored to a specific algorithm (NB in our study), the results are not so good for the rest of the classifiers, even when these classifiers belong to the same family (naive or semi-naive BNCs).

Mainly as a matter of interest, a more general BN classifier (with hill climbing as search algorithm), referred to as BNHC, has also been included. It provided lower results in terms of accuracy on average than AODE and HAODE, with a training time much larger than the rest of classifiers considered and a higher demand in terms of RAM memory. Even though the individual results (per dataset) provided by this classifier follow a more different pattern than the rest of semi-naive classifier, the global comparison places it in concordance with the rest.

A direct extension for future work consists in considering a wider range of discretization techniques, covering other families such as multivariate discretization methods, which despite being less efficient in terms of complexity, seem very promising regarding their properties.

Chapter 7

Non-disjoint discretization techniques

However beautiful the strategy, you should occasionally look at the results.
Sir Winston Churchill. (1874-1965)
British politician and statesman

Abstract

There is still lack of clarity about the best manner in which to handle numeric attributes when applying BNCs. In the previous chapter, both AODE and HODE's performance have shown to be robust towards the discretization method applied. However, all the discretization techniques taken into account so far formed non-overlapping intervals for a numeric attribute. We argue that the idea of non-disjoint discretization, already justified in NB classifiers, can also be profitably extended to AODE and HAODE, albeit with some variations; and our experimental results seem to support this hypothesis, specially for the latter.

7.1 Introduction

The discretization process entails grouping together consecutive continuous samples to form discrete groups of members, usually called bins or intervals. It is hence unavoidable to suffer from loss of information in this process, but still, the approximation provided can be more accurate than assuming unrealistic distributions.

However, when discretization is to be applied, many questions arise: to start with, which is the proper number of bins to form? How should the selection of the different cut-points be carried out? But also, should every sample belong to a single interval? That is, a decision must be made on the discretization method to apply. In which degree should we be concerned about this decision?

In this respect, Chapter 6 analyses the robustness of AODE and HAODE (along with other BNCs) regarding the discretization method. The conclusions in this study indicate that although the discretization method indeed matters when studying a particular dataset, it does not seem to be decisive when the aim is to compare a group of semi-naive BNCs over a standard group of datasets. Nevertheless, only disjoint discretization (DD) techniques have been taking into account so far. In Yang & Webb [2002], a novel non-disjoint discretization (NDD) technique is presented to cope with numeric attributes in NB by forming overlapping intervals. NDD forms overlapping intervals for a continuous attribute, always locating a value towards the middle of an interval to obtain more reliable probability estimations. Its use is based on the insight that while it is necessary to use a single discretization of each variable while classifying an instance, different discretizations can be applied when classifying different instances. The results show a clear improvement in NB over other DD methods. Compared to NB, AODE and HAODE could suffer more from creating a large number of intervals (from a variance increase), since their CPTs are formed by the combination of a couple of attributes (the class and the parent). It is credible that NDD could help us to alleviate this problem by allowing larger intervals to be formed without greatly increasing the bias.

The main contributions of this chapter are the following: to begin with, we redefine the original approach of NDD discretization for its use in AODE and HAODE, describing the corresponding modifications (Section 7.2). Furthermore, a new weighting system is included with the aim to decrease discretization bias. In Section 7.3, an experimental study compares the application of these NDD techniques in AODE and HAODE with the use of a traditional DD method: equal frequency discretization (EFD)[1]. This study includes comparisons in terms of accuracy, but mainly focuses in results detailing bias and variance discretization records. Finally, Section 7.4 provides our main conclusions from the study.

7.2 NDD adapted to AODE and HAODE

By dividing the ranges of numeric attributes into overlapping intervals in AODE and HAODE, we not only intend to reduce discretization bias [Yang & Webb, 2009] by always locating a value towards the middle of an interval and, in general, creating a larger number of intervals; but also maintaining discretization variance, since the number of samples from which the CPTs will be estimated should be similar.

The application of NDD to AODE involves discretizing the whole dataset into non-disjoint intervals before training the classifier, whereas in the case of HAODE, just the cases where a numeric attribute plays the role of super-parent will be discretized [Martínez *et al.*, 2012].

Furthermore, and for the reasons that we detail next, some changes are introduced to the original definition of NDD as specified in Yang & Webb [2002]:

1. A threshold is considered to mark the minimum frequency from which an atomic interval will not be merged with its neighbours. This should prevent us from increasing bias when sufficient samples are already provided. See figure 7.1 for an example on interval formation having each atomic interval frequency into account. Since it is possible the presence of multiple instances with the same value, the number of final samples per atomic attribute may vary, and it usually does[2].

[1]As deducted from previous chapters, this selection has not been made at random, since EF5 has shown to be the most beneficial for AODE.

[2]The way in which this is handled is the same for NDD and EF5, check WEKA's equal frequency discretization method for more details.

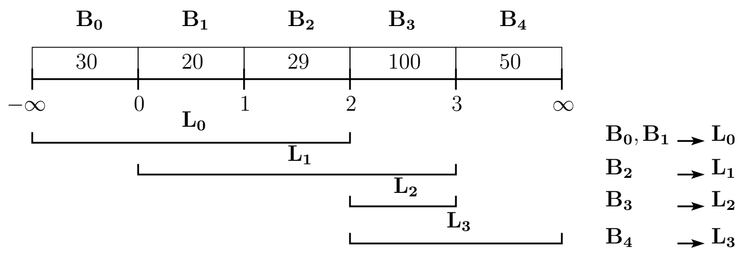
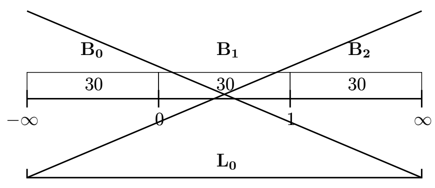


Figure 7.1: Example of NDD division, the minimum frequency to merge atomic intervals into a single label (L) is equal to 100. The labels selected when classifying samples belonging to atomic bins B_0, B_1, B_2, B_3 and B_4 are indicated at the bottom right corner.

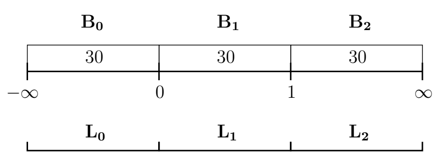
2. In the original definition of NDD, the interval size is equal to the interval number ($\approx \lfloor\sqrt{m}\rfloor$) with the aim to give equal importance to discretization bias and discretization variance reduction. Even though it provides very good results for NB, it is not the case for AODE or HAODE, where in general, a smaller number of intervals is desired. Both in AODE and HAODE, it is necessary to estimate the probability of an interval on one attribute conditioned by both an interval on another attribute and the class, whereas in NB it is necessary only to estimate the probability of an interval given the class. Previous experiments have shown that PD, proportional discretization tailored to NB where a number of $\lfloor\sqrt{m}\rfloor$ instances is selected, is not generally beneficial for AODE (see Section 6.2.4).

3. When the number of cut-points is lower than 3, then EFD will be kept. Figure 7.2 shows and example of this special case.

4. **Weighting importance or weighted NDD (wNDD)**: note that by using NDD as defined above, there are some numeric samples that fall within two or three labels. Given a numeric sample x_i discretized by NDD into the labels $L_1 = (a'_1, b'_1]$, $L_2 = (a'_2, b'_2]$ and $L_3 = (a'_3, b'_3]$ in training time; L_2 would be the final label assigned to another sample $x_j \in \mathbb{R}$, $x_j = x_i$ in classification time. The contribution of L_2 to the CPT will be greater (it is given more importance when training) than the contribution provided by the other two bins. This is carried



(a) It would be merged since the frequency is below the limit.



(b) EFD is kept.

Figure 7.2: Example of the special case when the number of cut-points is lower than 3. Even thought the sum is below the limit (100), EFD will be kept instead of merging.

out by the use of weights. There exist several forms in which these weights could be distributed, in this first approach we have adopted the simplest one (apart from uniform distribution, being equivalent to non-weighting). Since a single sample can be allocated at most in three atomic bins, the weight distribution could be set as 0.75 for the centred label and the rest equally divided into the other labels (if there is more than one)[1]. See Table 7.1 for more details. In AODE, the combination of weights when both the parent and the child involved in a CPT come from a joint discretization is carried out by multiplying its corresponding weights (so that the sum remains equal to one).

Figure 7.3 shows an example for a training instance I with two numeric attributes: X_0 and X_1. This instance is discretized using the NDD procedure indicated in Section 7.2, obtaining I_{NDD}. Hence, the value 3.5 for X_0 falls within three labels: L_0, L_1 and L_2 (specifically centred in L_1, that is why it is given the highest weight), whereas the value 2 for X_1 falls within labels L'_0 and L'_1 (centred in L'_1 in this case). These weights are then used to indicate the contribution of each pair of values when updating the CPTs[2]. When the same instance I were

[1]Further experiments have been carried out by slightly altering the weight assignment obtaining very similar results. This study has been performed using 3 atomic bins per interval, and we believe that this result may not be extrapolated to higher odd numbers.

[2]Note that in a multinomial distribution, the combination of values from an instance to be incorporated in a CPT contribute with a `unit`, whereas here we consider the con-

Table 7.1: Distribution of weights when using wNDD. w_0 corresponds to the weight to be assigned to the bin in the centre (i.e. label to be selected in classification time by wNDD).

	1 interval	2 intervals	3 intervals
w_0	1	0.75	0.75
w_1		0.25	0.125
w_2			0.125

to be classified (the class is missing), then I'_{NDD} would be used, where the centred labels for both attributes are considered. Then the MAP equation would be as follows:

$$argmax_{c\in\Omega_C}\left(\sum_{j=1,N(x_j)>q}^{n} p(c,L_j^*)\prod_{i=1,i\neq j}^{n} p(L_i^*|c,L_j^*)\right),$$

where L_j^*, is the centred label for X_j; and L_i^*, the centred label for X_i.

As NDD is dominated by sorting, no increase in the algorithm complexity order is induced.

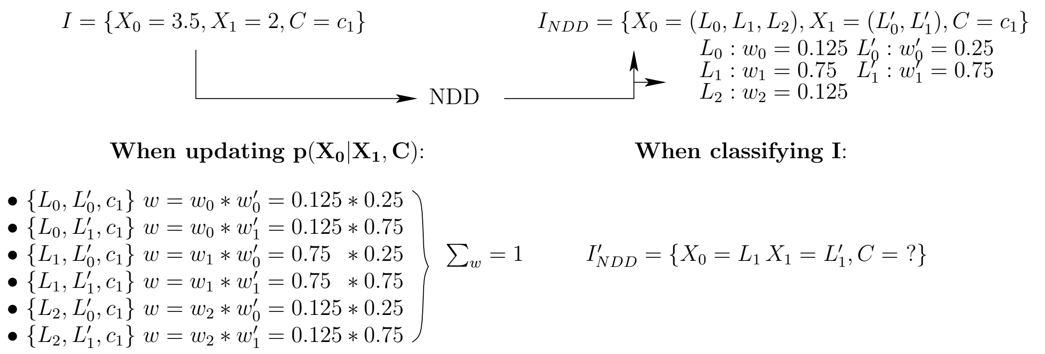

$I = \{X_0 = 3.5, X_1 = 2, C = c_1\}$
NDD
$I_{NDD} = \{X_0 = (L_0, L_1, L_2), X_1 = (L'_0, L'_1), C = c_1\}$
$L_0 : w_0 = 0.125$ $L'_0 : w'_0 = 0.25$
$L_1 : w_1 = 0.75$ $L'_1 : w'_1 = 0.75$
$L_2 : w_2 = 0.125$
When updating $\mathbf{p(X_0|X_1, C)}$:
When classifying $\mathbf{I}$:
• $\{L_0, L'_0, c_1\}$ $w = w_0 * w'_0 = 0.125 * 0.25$
• $\{L_0, L'_1, c_1\}$ $w = w_0 * w'_1 = 0.125 * 0.75$
• $\{L_1, L'_0, c_1\}$ $w = w_1 * w'_0 = 0.75 * 0.25$
• $\{L_1, L'_1, c_1\}$ $w = w_1 * w'_1 = 0.75 * 0.75$
• $\{L_2, L'_0, c_1\}$ $w = w_2 * w'_0 = 0.125 * 0.25$
• $\{L_2, L'_1, c_1\}$ $w = w_2 * w'_1 = 0.125 * 0.75$
$\sum_w = 1$
$I'_{NDD} = \{X_0 = L_1\, X_1 = L'_1, C = ?\}$

Figure 7.3: Example on how wNDD works in AODE: first of all, the instance is discretized using NDD and weights are assigned to every label. When training, instance I would contribute to the CPT for X_0 given X_1 and C as shown in the left hand side. If classifying I, only the main labels (so that the sample is in the centre) are considered.

tribution of the `weight` for each label (that always sums to one for each instance).

7.3 Experimental methodology and results

We run our experiments on 28 datasets from the UCI machine learning repository [Frank & Asuncion, 2010] and KDD archive [Hettich & Bay, 1999], listed in Table 7.2 (in increasing order of the number of instances). As in Yang & Webb [2002], this experimental suite comprises 3 parts. The first part is composed of all the UCI datasets used by Fayyad & Irani [1993] when publishing the entropy minimization heuristic discretization. The second part is composed of all the datasets with numeric attributes used by Domingos & Pazzani [1997] for studying NB classification. The third part is composed of larger datasets employed in Yang & Webb [2001].

Table 7.2: Main characteristics of the 28 hybrid datasets: number of predictive continuous variables ($\#C$), number of predictive discrete variables ($\#D$), number of classes (c) and number of instances (m).

Id	Datasets	$\#C$	$\#D$	c	m	Id	Datasets	$\#C$	$\#D$	c	m
1	labor-negotiations	8	8	2	57	15	annealing	6	32	6	898
2	echocardiogram	5	1	2	74	16	german	7	13	2	1000
3	iris	4	0	3	150	17	multiple-features	3	3	10	2000
4	hepatitis	6	13	2	155	18	hypothyroid	7	18	2	2163
5	wine-recognition	13	0	3	178	19	satimage	36	0	6	6435
6	sonar	60	0	2	208	20	musk	166	0	2	6598
7	glass-identification	9	0	3	214	21	pioneer-mobile-robot	29	7	57	9150
8	heart-disease	7	6	2	270	22	handwritten-digits	16	0	10	10992
9	liver-disorders	6	0	2	345	23	sign-language	8	0	3	12546
10	ionosphere	34	0	2	351	24	letter-recognition	16	0	26	20000
11	horse-colic	7	14	2	368	25	adult	6	8	2	48842
12	credit-screening	6	9	2	690	26	impums.la.99	20	40	13	88443
13	prima-indians-diabetes	8	0	2	768	27	census-income	8	33	2	299285
14	vehicle	18	0	4	846	28	forest-covertype	10	44	7	581012

We have adopted two pre-processing steps to begin with. This is in order to make the group of datasets uniform and suitable for all the classifiers considered in the comparison:

- Unsupervised filter to replace all the missing values with the modes and means from the existing data in the corresponding column.

- Unsupervised filter to remove useless attributes that do not vary at all or whose variation percentage is greater than 99% [1].

In order to evaluate the experimental results we employ two evaluation measures: accuracy and error in terms of bias and variance according to Kohavi & Wolpert [1996].

Once again, 5x2cv has been used to estimate accuracy. The bias-variance decomposition has been performed using the sub-sampled cross-validation procedure as specified by Webb & Conilione [2002].

As indicated above, the discretization technique selected as the basis for comparison is EF5, as it has shown to provide slightly better results for AODE compared to other DD techniques.

Since the final intervals formed in NDD will comprise at most 3 atomic bins[2], in order to provide a fair comparison with EF5, the initial number of atomic bins considered is 15. This means that the final bins (groups of three atomic bins) will be of approximately the same average size as the bins for EF5. The minimum frequency from which an atomic interval will not be merged with its neighbours will be 100 (approximately 30 per atomic bin[3]).

Table 7.3 shows the accuracy results obtained for AODE and HAODE using EF5, NDD and wNDD along with the sample standard deviation for each dataset. The bullet next to certain outputs (in NDD and wNDD) indicates that the corresponding result improves the output provided when EF5 is used. The circle, in turn, indicates a draw. These results lead us to think that the use of NDD or wNDD is competitive over EF5 (and by extension, other traditional DD techniques), especially for the former. Nevertheless, standard deviation is on average a bit higher for NDD and wNDD compared to EF5, although this difference is not statistically significant. This could indicate that EF5 is a bit more robust with respect to the income data, in spite of providing lower accuracy records.

[1]These two filters have been applied with the default settings provided by WEKA.

[2]In theory any odd number would be acceptable (the larger the better to allocate a sample in the middle of an interval), but for simplicity we take 3 as in Yang & Webb [2002].

[3]The value 30 has been selected motivated by the 30-sample rule-of-thumb, very recurrent in statistics. Still, further experiments were carried out with different values; although the results were not significantly different, the best values were obtained with 30 and $33.\overline{3}$.

Table 7.3: Results in terms of accuracy±sample standard deviation obtained for AODE and HAODE using EF5, NDD and wNDD.

	AODE			HAODE		
Id	EF5	NDD	wNDD	EF5	NDD	wNDD
1	93.3333±3.88	●94.3860±4.49	●94.0351±5.35	90.8772±8.59	●91.2281±9.76	●91.2281±9.48
2	68.9189±4.81	●72.9730±4.77	●72.1622±7.10	74.3243±4.64	72.9730±5.41	●76.7568±5.44
3	92.9333±2.74	●93.8667±2.20	●93.0667±1.97	95.8667±1.83	●96.0000±2.08	95.4667±1.80
4	82.1935±2.81	●82.9677±3.86	●82.4516±3.54	83.2258±2.23	82.0645±3.23	82.7097±3.09
5	96.4045±1.28	●96.8539±1.66	○96.4045±1.48	98.0899±0.93	○98.0899±0.76	97.8652±0.98
6	81.4423±3.63	●81.6346±4.19	80.7692±4.03	82.7885±3.91	82.5000±3.76	●84.6154±3.74
7	68.1308±5.07	●68.5047±4.06	●70.2804±3.76	69.1589±4.13	●69.5327±4.83	●70.0000±4.77
8	81.4815±2.42	●83.4815±2.59	●81.7037±1.60	81.0370±1.95	●81.5556±1.96	○81.0370±2.84
9	60.3478±3.47	●65.1014±3.46	●63.5942±2.76	62.0290±3.56	59.5942±3.81	61.1014±3.40
10	91.3390±2.19	89.4017±2.55	90.3134±2.42	92.2507±2.33	●92.7066±1.68	●92.4217±1.37
11	79.5652±1.23	●80.1087±1.94	●80.7609±1.18	65.6522±4.65	●66.3043±4.42	●66.4674±3.95
12	86.4638±1.23	●86.5797±0.95	●86.5507±1.18	80.7826±1.16	●80.0870±0.88	80.0870±1.14
13	75.2083±1.78	●75.5208±1.33	74.1927±1.65	75.6250±0.90	75.2344±0.94	75.0260±1.08
14	69.2199±1.14	68.3215±1.39	67.9433±1.58	73.3806±2.05	●73.5225±2.13	72.2695±2.25
15	87.9955±1.77	86.3474±1.65	●90.0668±1.07	82.9176±1.61	81.9822±2.81	82.5167±2.64
16	74.1600±1.08	●74.3800±0.93	●74.3400±1.19	73.7400±1.27	●74.7800±1.16	●74.1800±1.10
17	66.2600±1.22	●68.1700±1.26	●68.3600±1.31	69.1800±1.37	●69.9400±1.68	●70.6800±1.60
18	97.3000±0.21	●98.1979±0.27	●98.2548±0.22	98.1284±0.23	●98.3181±0.26	●98.3307±0.33
19	87.4219±0.57	●88.4444±0.40	●88.4444±0.40	83.9254±0.98	●85.9176±0.79	●85.9176±0.78
20	85.2743±0.85	●93.2404±0.32	●93.2555±0.30	83.5920±1.09	●87.5750±0.71	●87.5720±0.71
21	90.5268±0.47	●93.5432±0.86	●93.5016±0.87	89.1607±0.86	●94.3607±0.67	●94.3497±0.67
22	97.0287±0.17	96.8013±0.32	96.8013±0.32	97.1634±0.33	●97.6638±0.21	●97.6638±0.21
23	71.3678±0.70	●73.2680±0.51	●73.2855±0.51	66.3399±1.01	●67.1433±0.86	●67.1242±0.88
24	83.4580±0.21	●85.4120±0.37	●85.4720±0.37	84.5250±0.21	●88.1870±0.32	●88.2030±0.32
25	83.9347±0.25	●84.1677±0.29	●84.2771±0.29	84.0830±0.31	●83.9237±0.37	83.9892±0.35
26	92.3890±0.08	92.3854±0.08	●92.3928±0.08	87.0904±0.44	●87.7243±0.58	●87.7017±0.57
27	92.1766±0.09	●92.4165±0.07	●92.4171±0.07	93.4646±0.11	●93.6628±0.09	●93.6666±0.09
28	71.3988±0.11	●73.9682±0.09	●73.9682±0.09	69.9027±0.13	●70.8710±0.09	●70.8710±0.09
Av.	82.4169±1.62	●85.5873±1.67	●83.5381±1.67	81.7251±1.89	●82.2658±2.01	●82.4935±1.99

Table 7.4 shows the number of datasets for which discretizing with NDD obtained better, equal or worse performance compared to using EF5. These records are complemented by the results from the Wilcoxon signed-rank tests [Demšar, 2006], which compare every pair of algorithms considering the whole group of datasets. The first two columns depict the records when the samples are not weighted (i.e. weighted uniformly) according to the atomic bin to which they belong. In this case, NDD in AODE and HAODE is better at improving accuracy. The improvement is clear also as far as bias is concerned for HAODE and variance for AODE. However, this advantage is not as clear in terms of bias in AODE and variance in HAODE, although they still provide better records compared to EF5, no statistical difference is found. If we

Table 7.4: Comparisons in terms of win-draw-lose records and Wilcoxon tests for AODE and HAODE using EF5, NDD and wNDD.

w-t-l / Wilcoxon	non-weighted AODE NDD vs EF5	non-weighted HAODE NDD vs EF5	weighted AODE wNDD vs EF5	weighted HAODE wNDD vs EF5
Accuracy	23-0-5 / $< \mathbf{0.05}$	21-1-6 / $< \mathbf{0.05}$	22-1-5 / $< \mathbf{0.05}$	18-2-8 / $< \mathbf{0.05}$
Bias	14-3-11 / 0.2395	21-1-6 / $< \mathbf{0.05}$	15-3-10 / $< \mathbf{0.1}$(0.06)	22-0-6 / $< \mathbf{0.05}$
Variance	18-2-8 / $< \mathbf{0.05}$	14-4-10 / 0.3621	13-2-13 / 0.6	10-0-14 / 0.863

consider wNDD, the results are slightly better in terms of bias (specially for AODE), at the expense of variance and overall worsening. Hence, from now on in the chapter, we will just consider non-weighted NDD, although it is important to note that the increase in variance may have less effect on accuracy when larger data are provided.

Table 7.5 displays the average results in terms of accuracy, bias and variance obtained for the different classifiers, where NDD outperforms in every pair-to-pair comparison.

Table 7.5: Average results in terms of accuracy/bias/variance (best value in bold).

	AODE		HAODE	
	EF5	NDD	EF5	NDD
Accuracy	82.4169	**83.5873**	81.7251	**82.2658**
Bias	0.1298	**0.1250**	0.1348	**0.1275**
Variance	0.0395	**0.0355**	0.0440	**0.0435**

Figure 7.4 shows, on each circular graph, the individual bias and variance obtained over each dataset for AODE and HAODE with the two types of discretizations. Starting from the labelled radius where the results for the `labor` dataset are shown and going counter-clockwise, we find larger datasets (in terms of number of instances), being `covtype` (`forest-Covertype`) the largest one. Subfigures 7.4 (a) and 7.4 (b) depict the results in terms of bias, 7.4 (c) and 7.4 (d) in terms of variance and 7.4 (e) and 7.4 (f) the error

for AODE and HAODE respectively. Even though some differences are too small to be shown with precision, an overall tendency for NDD to enclose EF5 can be seen. But most importantly, the improvement of NDD over EF is more noticeable in the third and fourth quarter in the circle, indicating that NDD tends to perform better the more data we have. This last observation follows for all cases except for AODE in terms of variance, where NDD seems to have an advantage for small datasets.

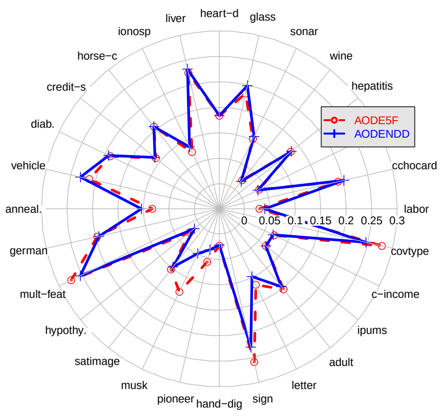
Note that execution time comparisons would show non-relevant information, since differences are minimum (same complexity order).

Hence, in light of these results one question arises: why does NDD seem to improve more pronouncedly AODE's variance and HAODE's bias compared to applying equal frequency? The difference between the two classifiers lies in the "double use" (in parents and children nodes) of NDD in AODE, which seems to help in reducing variance at the expense of a bias sacrifice.

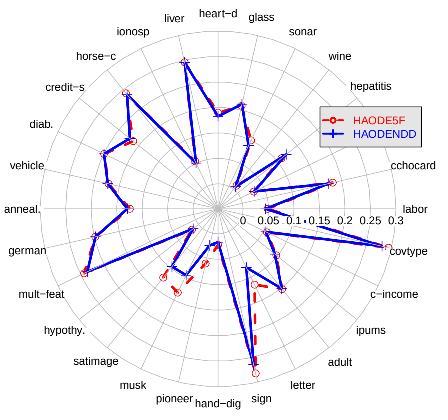
In this study, even though there is a slight improvement of HAODE over AODE (16-0-12 in terms of accuracy, see Table 7.3), this is not as striking as in the original study in Chapter 4, and this difference even shifts to 13-1-14 when NDD is applied. We believe this fact might be motivated by two reasons:

- HAODE aims to avoid information loss by resorting to the use of discretization only when necessary for the super-parents. However, that implies that Gaussian distributions are assumed in some cases, which can be a handicap if the real distributions in data are not Gaussians.

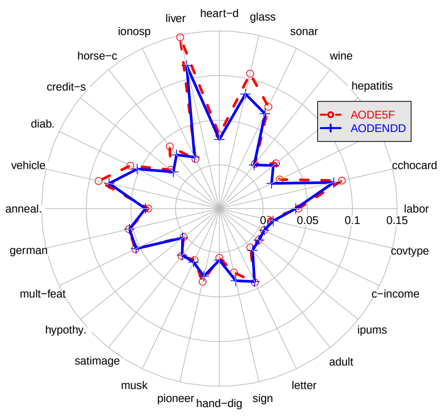
- In general, we should prefer high-bias, low-variance classifiers when the data are sparse; and low-bias, high-variance classifiers when data are numerous. Since we are now dealing with larger datasets, we could also deduce that HAODE is more robust in small ones and AODE in larger ones, unless the normality condition is satisfied.



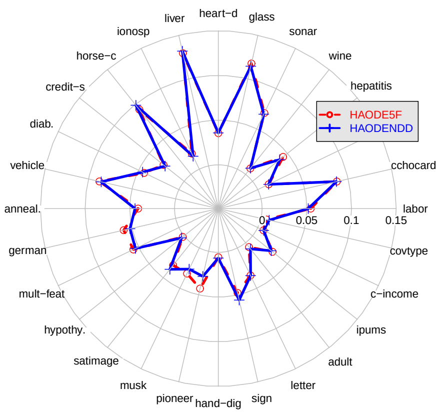
(a) Bias AODE EF5 vs NDD



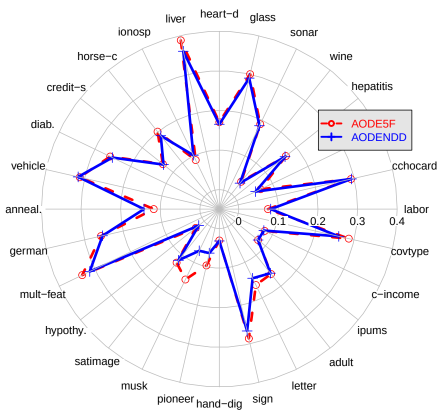
(b) Bias HAODE EF5 vs NDD



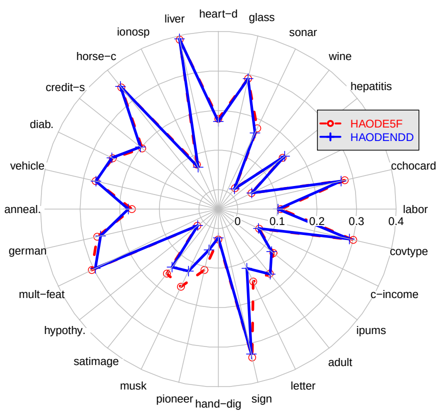
(c) Variance AODE EF5 vs NDD



(d) Variance HAODE EF5 vs NDD



(e) Error AODE EF5 vs NDD



(f) Error HAODE EF5 vs NDD

Figure 7.4: Graphical representation of individual results in terms of bias, variance and error for AODE and HAODE, using EF5 and NDD.

7.4 Conclusions and future work

In this chapter, we have studied the impact of applying non-disjoint discretization for AODE and HAODE classifiers compared to traditional disjoint discretization techniques. We have chosen equal frequency division to represent the latter, as it was previously shown to provide better results among the most common disjoint discretization methods (EF, equal width division, MDL, etc.).

We have introduced some modifications to the original definition of NDD [Yang & Webb, 2002] in order to fit into AODE and HAODE's context, as a smaller number of bins is usually desired compared to NB to avoid increasing variance.

Furthermore, a new weighting system has been introduced at the counting process in order to increase the importance given to the bins created by NDD where samples are placed in the middle; which provided better results in terms of bias but worse overall records.

The results have been analysed in terms of accuracy, bias and variance obtaining the following conclusions:

- In general terms, an overall improvement is found for the two classifiers (AODE and HAODE) when NDD is used. Statistical differences according to the Wilcoxon test are found for both classifiers as far as accuracy is concerned.

- If we analyse the error decomposition in terms of bias and variance, we observe better results at all times when using NDD, but this improvement is more marked for HAODE in terms of bias, and AODE in terms of variance.

The most important conclusion though, is the fact that whereas some of the most common disjoint discretization techniques have failed to demonstrate consistent improvement relative to alternatives, non-disjoint discretization demonstrates better win/draw/loss records and significant overall improvement.

Moreover, we believe that the positive results observed in AODE are a good motivation to think that the beneficial properties of NDD will be strengthen when applied to Aggregating n-dependence estimators (AnDE) [Webb *et al.*, 2012] for values of n greater or equal to 2 (since when $n = 1$ it is equivalent to AODE).

One drawback of NDD is that it requires the user to select additional parameters apart from the number of bins to form (such as in equal frequency division), also the number of atomic bins per operational interval and the minimum frequency per interval must be chosen.

In the future, we find it of major interest to include a test bed of higher dimensional datasets, to investigate whether the bias/variance values behave similarly or even better, specially for the weighted approach, as indicated in light of the current results.

Part IV

Domains of competence of the semi-naive BNCs

Chapter 8

Domains of competence of the semi-naive BNCs

Crude classifications and false generalizations are the curse of organized life.
George Bernard Shaw. (1856 - 1950)
Irish playwright

Abstract
The motivation of this chapter comes from observing the recent tendency to assert that rather than a unique and globally superior classifier, there exist local winners. This idea can also be extracted from the experiments carried out on Chapters 3 to 7. Hence, the proposal of new classifiers can be seen as an aim to cover new parcels or even to compete with those previously assigned to others. The complexity measures for supervised classification have been designed to define these parcels. In this chapter, we want to discover which type of datasets, defined by certain range values of the aforementioned complexity measures, fits for some of the semi-naive BNCs considered along this thesis.

8.1 Introduction

The use of complexity measures (CMs) for supervised classification has received increasingly attention since their formal definition in Ho & Basu [2002]. Many subsequent studies have applied these measures to find out the domains of competence of different classifiers [Bernadó-Mansilla & Ho, 2004, 2005; Lu-

Type	Id.	Name	Simpler if...
Overlaps in the feature values from different classes	F1	Maximum Fisher's discriminant ratio	+
	F1v	Directional-vector maximum Fisher's discriminant ratio	+
	F2	Overlap of the per-class bounding boxes	−
	F3	Maximum (individual) feature efficiency	+
	F4	Collective feature efficiency	+
Measures of class separability	L1	Minimized sum of the error distance of a linear classifier	−
	L2	Training error of a linear classifier	−
	N1	Fraction of points on the class boundary	−
	N2	Ratio of average intra/inter class nearest neighbor distance	−
	N3	Leave-one-out error rate of the one-nearest neighbor classifier	−
Measures of geometry, topology, and density of manifolds	L3	Nonlinearity of a linear classifier	−
	N4	Nonlinearity of the one-nearest neighbor classifier	−
	T1	Fraction of maximum covering spheres	−
	T2	Average number of points per dimension	+

Table 8.1: Summary of CMs for supervised classification.

engo & Herrera, 2009, 2010a,c; Sánchez *et al.*, 2007]. Other works have also attempted to generalize these measures to problems with multiple classes [Mollineda *et al.*, 2005; Orriols-Puig *et al.*, 2010], as the original definition only covers binary-class problems (more details in Section 2.3).

However, there is no work done in this sense in relation to semi-naive BNCs and neither for discrete domains, since these CMs have been applied to exclusively numeric domains.

Hence, our objective in this chapter is to explore the behaviour of some of the semi-naive BNCs according to the values of the CMs in literature for a particular group of datasets. Since the natural domain of BNCs comprises exclusively discrete attributes, we want to analyse as well, the descriptive power of the CMs on discrete domains. For these purposes, we use the measures summarized in Table 8.1, where as a novelty, a new column is added to indicate the tendency that a particular measure may follow according to its definition, in order to reflect more simplicity on the problem it applies. Note that the symbol + indicates more simplicity as the value of the corresponding complexity measure increases, and − as it decreases.

The rest of the chapter is divided as follows: Section 8.2 provides a study of the individual characterization of NB and AODE (in discrete and continuous domains) based on several CMs. Section 8.3 includes a comparison of the values of the CMs on continuous datasets and its version after discretization. And finally, in Section 8.4, a meta-classifier to predict the best semi-naive BNC based on some of these CMs is proposed and empirically tested.

8.2 Domains of competence of NB and AODE

In this section we study the behaviour, in terms on training and test accuracy, of NB and AODE, according to the values of the different measures. In the first place, we will work on discrete domains for both classifiers, resorting to unsupervised discretization; and secondly, we will carry out the same study on the same group of datasets but recovering its continuous original values, that will be handled with Gaussian distributions by both classifiers (GNB and GAODE).

8.2.1 Discrete domains

Some of the CMs have been originally defined for numeric values. Still, nominal attributes can be mapped into integer numbers for all the calculations, assuming though an non-existent order between the labels.

8.2.1.1 Naive Bayes - EF5

It is not easy to determine when a particular classification method will perform successfully on a given dataset. It is well known that accuracy on training is not a good estimator, since it usually overfits data. However, this overfitting can be more severe in some classifiers than others.

Figure 8.1 shows the differences in terms of accuracy on training and test for a group of datasets ordered in ascending accuracy in training. 5x2cv has been used for the evaluation. Even though overfitting is not generally too high compared to other more complex classifiers, it affects some datasets more than others.

As test bed for this and the following experiments in this chapter, we are using the group of 26 numeric datasets already presented in Table 4.1. Since most of the CMs are only defined to deal with datasets with two class labels, we have created several binary datasets from each dataset with more than 2 class labels, specifically, as many as the total number of class labels per dataset (by following the strategy known as one-against-all). Hence, we work with a total of 157 datasets with two class labels. Additionally, we

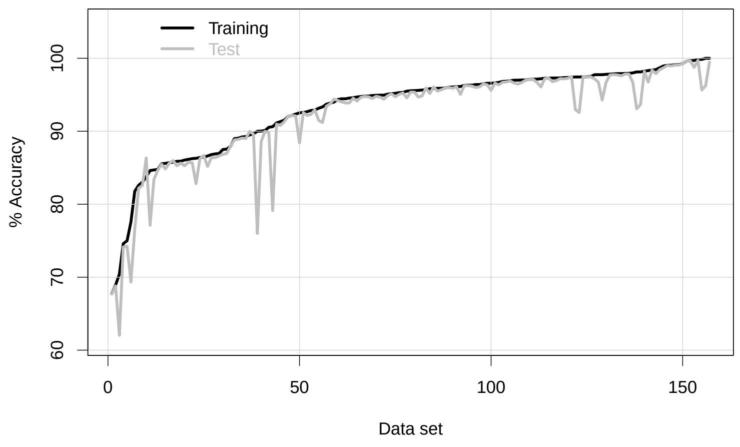


Figure 8.1: Datasets in increasing order of accuracy in training for NB.

discretize them applying unsupervised equal frequency discretization with 5 bins (EF5).

Now, we want to find the range of the measures for which NB shows good or bad behaviour. To accomplish it we calculate the values for the 14 CMs in Table 8.1 on each dataset. Afterwards, we could simply observe the behaviour of the datasets (in terms of accuracy in training and test) when they are ordered according to the value of a particular complexity measure, and the ranges of good/bad behaviour could be obtained directly in light of the graphs. Even though in principle, it seems to be the most accurate method, it is tedious and subject to lack of rigour, since the creation of the different intervals would follow different criteria.

Hence, we consider the method suggested in Luengo & Herrera [2010b], in which the authors propose an automatic algorithm to extract the ranges for good and bad behaviour depending on the values of a single complexity measure. In particular, an **interval of good behaviour** is considered when both the differences in average training and test accuracy with the global average are lower than a fixed value. An **interval of bad behaviour** is considered when any of the differences in average training and test accuracy with the global average are greater than a fixed value, or overfitting is observed.

Let $U = \{u_1, u_2, \ldots, u_m\}$ be a list of paired training and test accuracy values for m different datasets ($m = 157$ in our case). Following the notation in Luengo & Herrera [2010b], then u_i^{tra} is the training accuracy value associated to the dataset u_i and u_i^{tst} the test accuracy for u_i. Besides, every u_i has associated a value for the complexity measure considered.

Two types of elements are then identified: points of good and bad behaviour, that take into account the presence of overlearning; and the intervals of good and bad behaviour, that consider differences in their average accuracy $\bar{V}$ with respect to the global one $\bar{U}$:

- A **good behaviour point** u_i is such that:
    1. $u_i^{tra} - u_i^{tst} \leq overLearningDifference$; and
    2. $u_i^{tra} \geq minGoodTraining$.
- A **bad behaviour point** u_i is such that:
    1. $u_i^{tra} - u_i^{tst} > overLearningDifference$; or
    2. $\bar{U}^{tra} - u_i^{tra} \geq minBadGlobalDifference$.
- An **interval of good behaviour** V is such that:
    1. $\bar{V}^{tra} - \bar{U}^{tra} \geq inTrainDifference$; and
    2. $\bar{V}^{tst} - \bar{U}^{tst} \geq inTestDifference$.
- An **interval of bad behaviour** V is such that:
    1. $\bar{U}^{tra} - \bar{V}^{tra} \geq minBadGlobalDifference$; or
    2. $(\bar{V}^{tra} - \bar{V}^{tst}) - (\bar{U}^{tra} - \bar{U}^{tst}) \geq overLearningDifference$; or
    3. $\bar{U}^{tst} - \bar{V}^{tst} \geq inTestDifference$.

The different intervals of good and bad behaviour are created then following Algorithm 8.1. The functions invoked in the algorithm are defined as follows[1]:

[1]More details on how these intervals are automatically extracted and the value of the different parameters, can be found in Luengo & Herrera [2010b]. The same parameters

- $nextImportantGoodPoint(u_i, U)$ and $nextImportantBadPoint(u_i, U)$: return the next *good/bad behaviour point* respectively from u_i, or -1 if none can be found.

- $extendGoodInterval(pos, U)$ and $extendBadInterval(pos, U)$: return an *interval of good/bad behaviour* respectively from u_{pos}.

- $mergeOverlappedIntervals(I)$: removes those intervals that are included in others. Besides, it merges overlapped intervals or intervals that are separated by a maximum gap of 5 points, provided that the new merged interval satisfies the previous definitions of good or bad behaviour.

Table 8.2 shows the domains of the different CMs on the group of 157 discrete datasets considered. We have decided not to normalize these ranges to preserve as much genuine information as possible. The calculations of the different measures have been obtained with the data complexity library in C++ [Orriols-Puig *et al.*, 2010].

Figure 8.2 shows the datasets organised according to the values obtained for different CMs. A total of 6 CMs have been selected for this study, specifically, those whose graph representations seems to provide better patterns when the examples are placed in ascending order of the corresponding measure. In Subfigures 8.2 (a) to (f), the intervals of good and bad behaviour, given by the algorithm indicated above, have been marked. The lower X-axis indicates the number of datasets, whereas the upper one marks the value of the complexity measure for the interval's boundaries[1]. Note that if the intervals were made directly based on the visual representation they would have been selected differently. It would not be an easy task though, while the

have been considered here except for $inTrainDifference = 3$ and $inTestDifference = 3.5$, since the range of the accuracy results obtained for NB-EF5 is narrower (no smaller than 60) than the results provided by the fuzzy rule based classification system used as case study by the authors.

[1]Since the datasets are placed along the X-axis uniformly distributed and not according to the values of the CM, the marks in the upper X-axis are not lineally distributed. There might be several datasets with the same value for a particular CM for example.

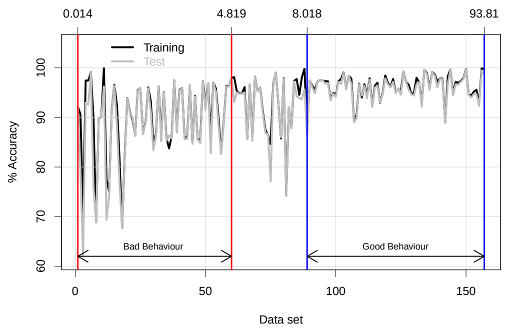
Algorithm 8.1: Interval automatic extraction method

```
Input: Dataset with values $U = \{u_1, u_2, \ldots, u_m\}$ sorted by a
       particular CM; $\bar{U}^{tra}$; $\bar{U}^{tst}$.
Output: Intervals of good and bad behaviour, $G$ and $B$ respectively.
1  $G = \{\}, B = \{\}, i = 0$;
2  while $i < m$ do
3      $pos = nextImportantGoodPoint(i)$;
4      if $pos \neq -1$ then
5          $interval = extendGoodInterval(pos, U)$;
6          $G = G \cup \{interval\}$;
7          $i = Up\{interval\}$
8      end
9  end
10 i=0;
11 while $i < m$ do
12     $pos = nextImportantBadPoint(i)$;
13     if $pos \neq -1$ then
14         $interval = extendBadInterval(pos, U)$;
15         $G = G \cup \{interval\}$;
16         $i = Up\{interval\}$
17     end
18 end
19 $G = mergeOverlappedIntervals(G)$;
20 $B = mergeOverlappedIntervals(B)$;
21 return $\{G,B\}$
```

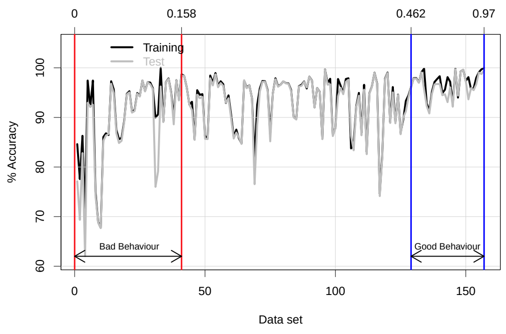
Table 8.2: Domains of the different CMs on the group of 157 datasets created from Table 4.1 after discretizing using EF5 (discrete datasets).

Domains	
$F1 \in [0.00082, 0.475]$	$N1 \in [0.004, 0.571]$
$F1v \in [0.008, 93.81]$	$N2 \in [0.014, 0.971]$
$F2 \in [0, 1]$	$N3 \in [0.000866, 0.435]$
$F3 \in [0, 0.97]$	$L3 \in [0, 0.5]$
$F4 \in [0, 1]$	$N4 \in [0, 0.5]$
$L1 \in [0.012, 2.706]$	$T1 \in [0.987, 1]$
$L2 \in [0.0005, 0.42]$	$T2 \in [3.467, 1250]$

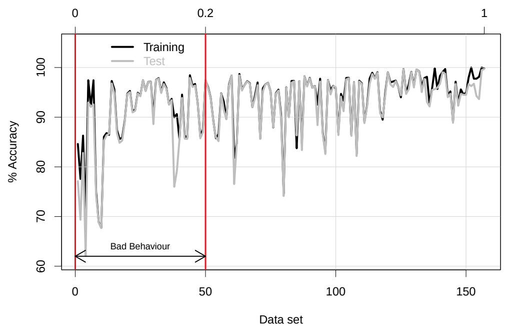
automatic method provides homogeneous results, and (probably) a higher generalization capability.



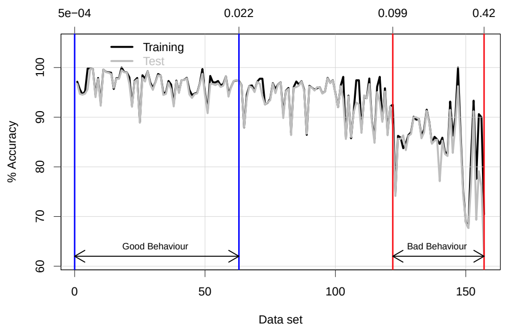
(a) NB F1v



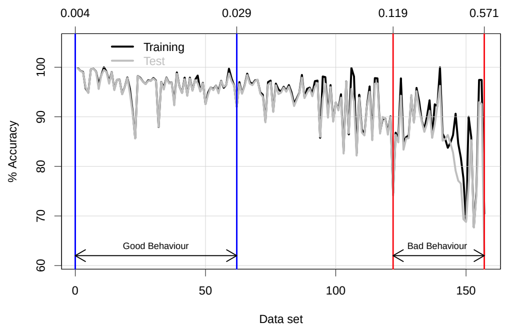
(b) NB F3



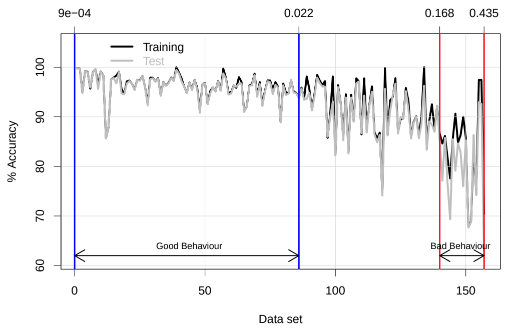
(c) NB F4



(d) NB L2



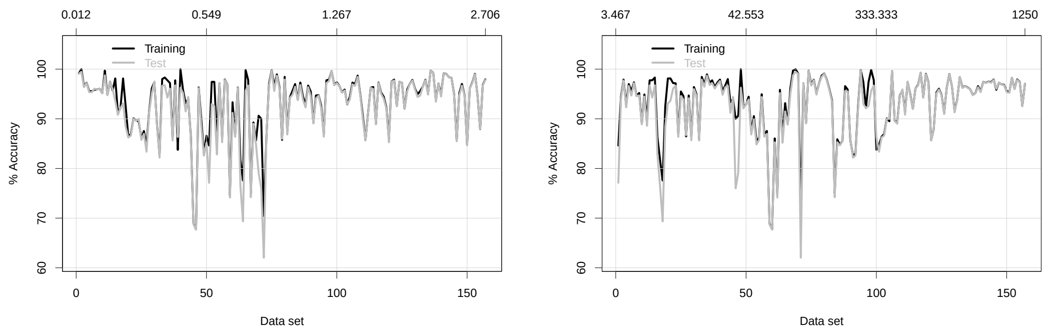
(e) NB N1



(f) NB N3

Figure 8.2: Characterization of the group of datasets into good or bad behaviour for NB according to increasing order of 6 CMs individually: F1v, F3, F4, L2, N1 and N3.

The tendency in these graphs follows, in greater or leaser degree, the pattern indicated in the last column in Table 8.1, i.e., for F1v, F3 and F4, the larger the values of the measures the better the behaviour of the classifier for these datasets (since they are supposed to become simpler to classify), and the opposite for L2, N1 and N3. Figure 8.3, in turn, shows the graphs obtained for L1 and T2, where it is difficult to observe a coherent pattern. Also T1 has been discarded in advance, since the values of T1 for most of the datasets are equal to 1.



(a) NB L1 (b) NB T2

Figure 8.3: Examples organised in increasing order of L1 and T2, for which patterns are not clearly identified.

Table 8.3 depicts the statistics of these intervals of good and bad behaviour obtained with the automatic method, providing more detailed quantitative information. In the first line, the global average accuracy on training and test (93.19% and 92.32% respectively), along with their standard deviations are shown. Those intervals characterizing datasets with good behaviour have been identified with the name Ri+, whereas those characterizing datasets with bad behaviour are identified with the name Ri- (first column). The percentage of datasets (out of the 157) that falls into a particular interval is indicated in the third column, %Support. Both average accuracy on training and test for each interval are shown, as well as the differences with the corresponding total average accuracy on training and test.

The percentages of examples supporting these intervals (43.31%, 17.83%,

Table 8.3: Rules for NB from the intervals automatically obtained.

	Training accuracy=93.1912 ± 6.1753				Test accuracy=92.3230 ± 6.8575	
Good behaviour rules						
Id.	Interval	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
R1+	$F1v \in [8.018, 93.81]$	43.3121	96.3806	3.1893	96.0235	3.7005
R2+	$F3 \in [0.462, 0.97]$	17.8344	97.1264	3.9351	96.2535	3.9304
R3+	$L2 \in [0.0005, 0.022]$	40.7643	96.7970	3.6058	96.4228	4.0997
R4+	$N1 \in [0.0040, 0.029]$	40.1274	96.5084	3.3172	96.3237	4.0006
R5+	$N3 \in [0.0009, 0.022]$	55.4140	96.2174	3.0261	95.9946	3.6715
Bad behaviour rules						
Id.	Interval	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
R1-	$F1v \in [0.014, 4.819]$	38.2166	89.9146	−3.2767	88.4610	−3.8620
R2-	$F3 \in [0.0, 0.158]$	26.7516	90.5030	−2.6883	88.6962	−3.6268
R3-	$F4 \in [0, 0.2]$	31.8471	90.3038	−2.8875	88.7971	−3.5260
R4-	$L2 \in [0.099, 0.42]$	22.2930	85.0267	−8.1646	82.9197	−9.4033
R5-	$N1 \in [0.119, 0.571]$	22.9299	86.6363	−6.5550	84.1298	−8.1932
R6-	$N3 \in [0.168, 0.435]$	10.8280	83.1718	−10.0195	79.2170	−13.1061

40.76%, ...) indicate that it is possible to characterize a wide range of datasets and to obtain significant differences in accuracy. However, the ranges obtained from different CMs may have overlapping examples, and in fact they do. Therefore, in order to obtain a description of the behaviour of NB based on disjoint intervals for the different variables we will create new general rules based on the combination of the "positive" and "negative" rules presented above.

Similarly to Luengo & Herrera [2010b], a positive rule disjunction (PRD) will be formed through the disjunction of all the "positive" rules, and a negative rule disjunction (NRD) with the disjunction of all the "negative" ones. Since these two groups may still overlap, and our aim is to obtain disjoint groups of good and bad behaviour; we also consider the following groups:

- PRD ∧ NRD: formed by those datasets that fall into both groups.
- PRD ∧ ¬NRD: formed by those datasets that accomplish at least one of the rules in PRD but none of the rules in NRD.
- NRD ∧ ¬PRD: formed by those datasets that accomplish at least one of the rules in NRD but none of the rules in PRD.

- ¬NRD ∧ ¬PRD: formed by those datasets that do not fall in any of the groups, i.e., datasets that can not be characterized by any of the CMs considered.

Table 8.4 shows the statistics of this new group of rules. Note that only 3 datasets (1.91% of support) are not characterized by the value of any of the CMs considered. Note that if the supports for the four combinations previously indicated (last four lines in Table 8.4) are summed, the result is the 100% of the datasets, naturally.

Table 8.4: Disjunction and intersection rules from all simple rules for NB.

Id.	If	Then (behaviour)	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
PRD	R1+ or R2+ or R3+ or R4+ or R5+	good	68.1529	96.1026	2.9114	95.7190	3.3959
NRD	R1- or R2- or R3- or R4- or R5- or R6-	bad	59.2357	91.3471	-1.8441	90.2215	-2.1016
PRD ∧ NRD	PRD and NRD	good	29.2994	96.0005	2.8093	95.7661	3.4431
PRD ∧ ¬NRD	PRD and ¬NRD	good	38.8535	96.1796	2.9884	95.6834	3.3604
NRD ∧ ¬PRD	NRD and ¬PRD	bad	29.9363	86.7927	-6.3985	84.7947	-7.5283
¬NRD ∧ ¬PRD	¬(PRD or NRD)	bad	1.9108	89.5962	-3.5951	89.1445	-3.1785

Finally, Figure 8.4 shows the two regions of good and bad behaviour obtained for NB. On the left hand side, those datasets characterized as good behaviour are displayed (with no particular order). This covers rules: (PRD ∧ NRD) and (PRD ∧ ¬NRD), with average accuracy equal to 96.1026 ± 2.58 on training and 95.719 ± 2.54 on test. On the right hand side, in turn, the datasets characterized as bad behaviour are shown. This covers the following rules: (NRD ∧ ¬PRD) and (¬NRD ∧ ¬PRD) in this case, with average accuracy equal to 86.9609 ± 7 and 85.0557 ± 7.53 on test.

Appendix A includes several graphs showing bivariate relationships between some of the CMs on NB.

8.2.1.2 AODE - EF5

In this section we perform a similar study on AODE, on the same group of 157 datasets discretized using EF5. Figure 8.5 shows these datasets in

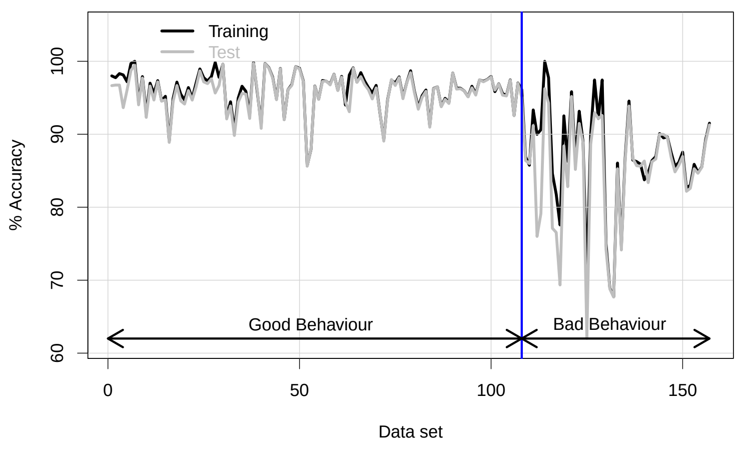


Figure 8.4: NB characterization in terms of good and bad behaviour from the disjunction and intersection rules of 6 CMs: F1v, F3, F4, L2, N1 and N3.

increasing order of training accuracy. We now see larger overfitting than in NB, as there is a higher difference between training and test accuracy values.

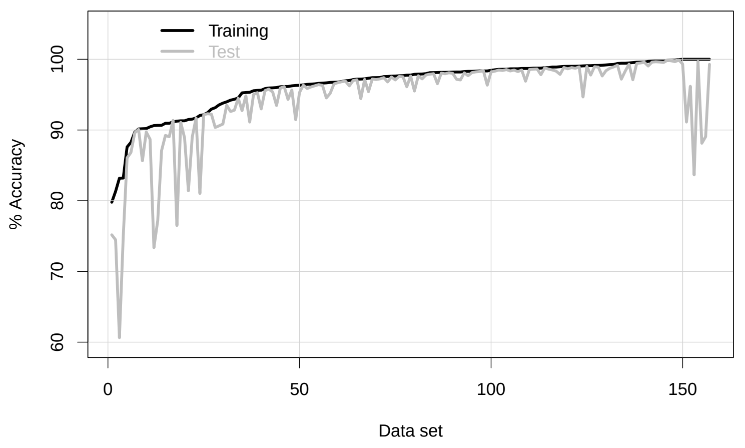


Figure 8.5: Datasets in increasing order of accuracy in training for AODE.

The parameters for the intervals' creation have been maintained here except for $inTrainDifference = 2$ and $inTestDifference = 3$, since the range of the accuracy results obtained is different (standard deviations are lower than in NB, specially in training accuracy). The results of the intervals created are graphically displayed on Figure 8.6. Note that the same group of CMs than for NB has been selected, since they provide the clearest patterns also for AODE.

The detailed quantitative information of all these intervals is shown in Table 8.5, and the disjunction and intersection rules are summarized in Table 8.6.

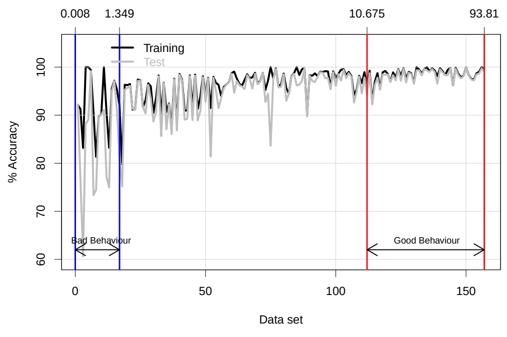
Table 8.5: Rules for AODE from the intervals automatically obtained.

	Training accuracy=96.4669 ± 3.7672			Test accuracy=94.7236 ± 6.1236		
Good behaviour rules						
Id.	Interval	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
R1+	$F1v \in [10.675, 93.81]$	28.6624	98.5685	2.1016	98.1840	3.4604
R2+	$F3 \in [0.462, 0.97]$	17.8344	98.9906	2.5237	98.3320	3.6084
R3+	$F4 \in [0.65, 1.0]$	28.0250	98.7006	2.2338	97.8829	3.1593
R4+	$L2 \in [0.0005, 0.032]$	50.9554	98.4885	2.0217	98.0002	3.2766
R5+	$N1 \in [0.0040, 0.027]$	38.8535	98.5087	2.0418	98.3076	3.5840
R6+	$N3 \in [0.0009, 0.02]$	52.8662	98.3675	1.9006	98.1123	3.3887
Bad behaviour rules						
Id.	Interval	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
R1-	$F1v \in [0.0080, 1.349]$	11.4650	91.7540	−4.7128	84.2153	−10.5083
R2-	$F3 \in [0, 0.2]$	33.1210	95.3222	−1.1447	91.6170	−3.1066
R3-	$F4 \in [0.0, 0.285]$	40.7643	94.9078	−1.5591	91.6761	−3.0475
R5-	$L2 \in [0.13, 0.42]$	11.4650	91.1862	−5.2806	82.7101	−12.0135
R4-	$N1 \in [0.21, 0.571]$	12.7388	91.3592	−5.1076	82.6454	−12.0782
R6-	$N3 \in [0.119, 0.435]$	15.2866	91.1543	−5.3126	83.6289	−11.0947

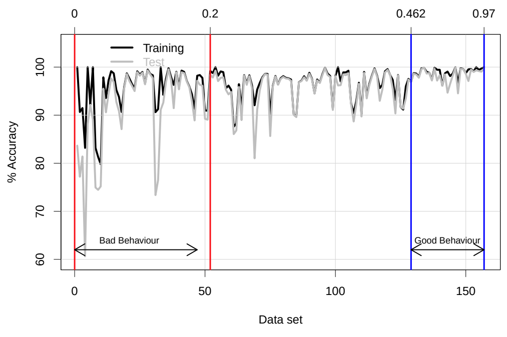
Table 8.6: Disjunction and intersection rules from all simple rules for AODE.

Id.	If	Then (behaviour)	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
PRD	R1+ or R2+ or R3+ or R4+ or R5+	good	69.4267	98.1488	1.6820	97.5593	2.8357
NRD	R1- or R2- or R3- or R4- or R5- or R6-	bad	52.2293	95.2300	-1.2369	92.3711	-2.3525
PRD ∧ NRD	PRD and NRD	good	26.1146	98.0156	1.5487	97.2134	2.4898
PRD ∧ ¬NRD	PRD and ¬NRD	good	43.3121	98.2292	1.7623	97.7679	3.0443
NRD ∧ ¬PRD	NRD and ¬PRD	bad	26.1146	92.4443	-4.0225	87.5289	-7.1947
¬NRD ∧ ¬PRD	¬(PRD or NRD)	bad	4.4586	93.8368	-2.6301	92.7078	-2.0158

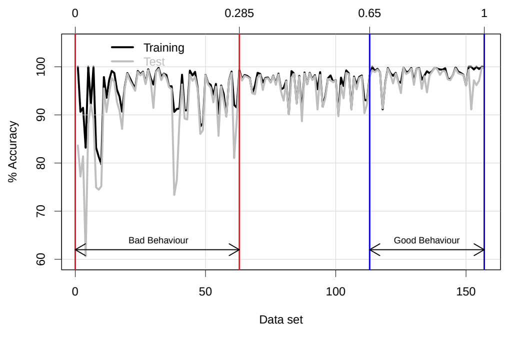
Figure 8.7 displays the final characterization for AODE. Average accu-



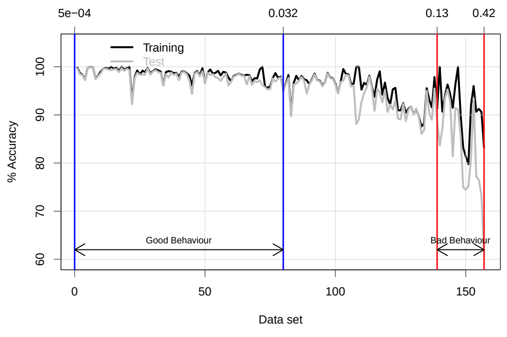
(a) AODE F1v



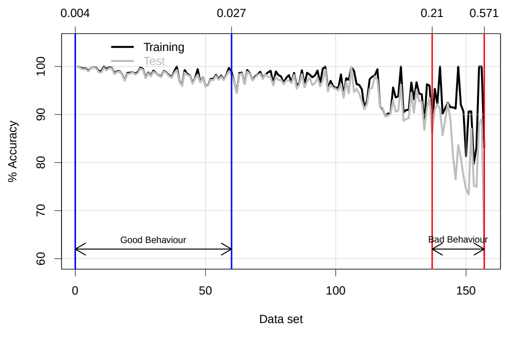
(b) AODE F3



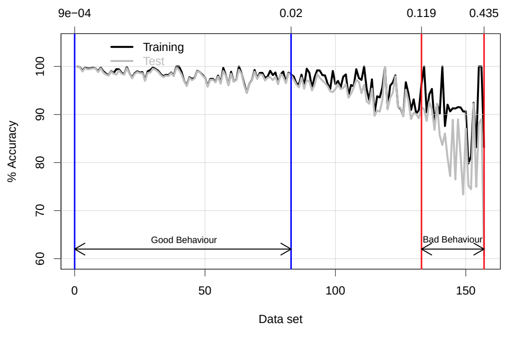
(c) AODE F4



(d) AODE L2



(e) AODE N1



(f) AODE N3

Figure 8.6: Characterization of the group of datasets into good or bad behaviour for AODE according to increasing order of 6 CMs individually: F1v, F3, F4, L2, N1 and N3.

racy obtained on training is equal to 98.1488 ± 1.48 for good behaviour and 92.6474 ± 4.52 for bad behaviour; whereas average accuracy on test is equal to 97.5593 ± 1.67 for good behaviour and 88.2842 ± 4.54 for bad behaviour.

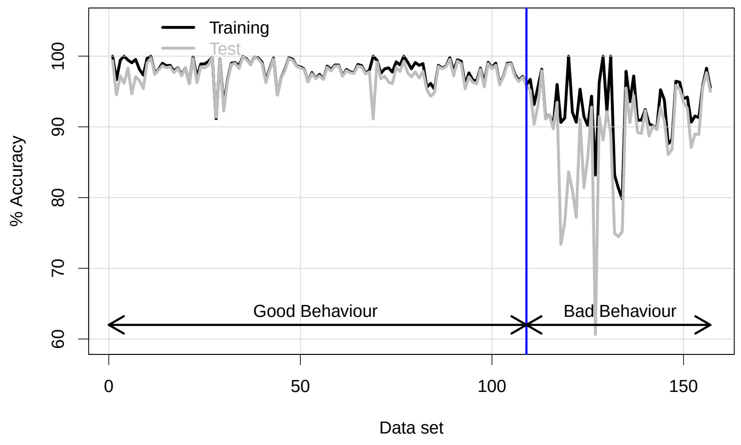


Figure 8.7: AODE characterization in terms of good and bad behaviour from the disjunction and intersection rules of 6 CMs: F1v, F3, F4, L2, N1 and N3.

8.2.2 Continuous domains

Whereas for discrete domains T1 was discarded, given that it practically did not vary for the different datasets; in continuous domains we encounter that neither L3 does, as it is approximately equal to 0.5 in most of the cases. Furthermore, also F2 is discarded, since its results are very close to 0 in many cases; and T1 remains uninformative.

Table 8.7 shows the domains of the different CMs for the group of 157 datasets considered in continuous domains.

8.2.2.1 Gaussian naive Bayes

Figure 8.8 shows these datasets in increasing order of training accuracy for NB in continuous domains using Gaussian distributions.

Table 8.7: Domains of the different CMs for the group of 157 datasets created from Table 4.1 (continuous datasets).

Domains	
$F1 \in [0.001, 65.2]$	$N1 \in [0.00071, 0.574]$
$F1v \in [0.000068, 132.917]$	$N2 \in [0.004, 0.922]$
$F2 \in [0, 1]$	$N3 \in [0, 0.374]$
$F3 \in [0, 0.999]$	$L3 \in [0.001, 0.5]$
$F4 \in [0, 1]$	$N4 \in [0, 0.494]$
$L1 \in [0.01, 0.895]$	$T1 \in [0.313, 1]$
$L2 \in [0.002, 0.461]$	$T2 \in [3.467, 1250]$

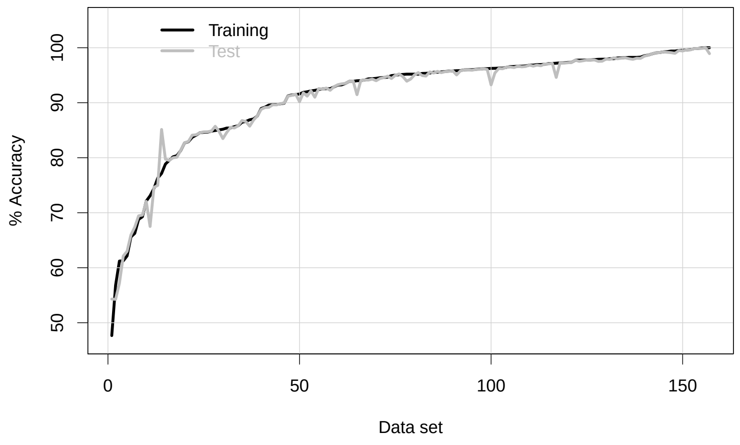
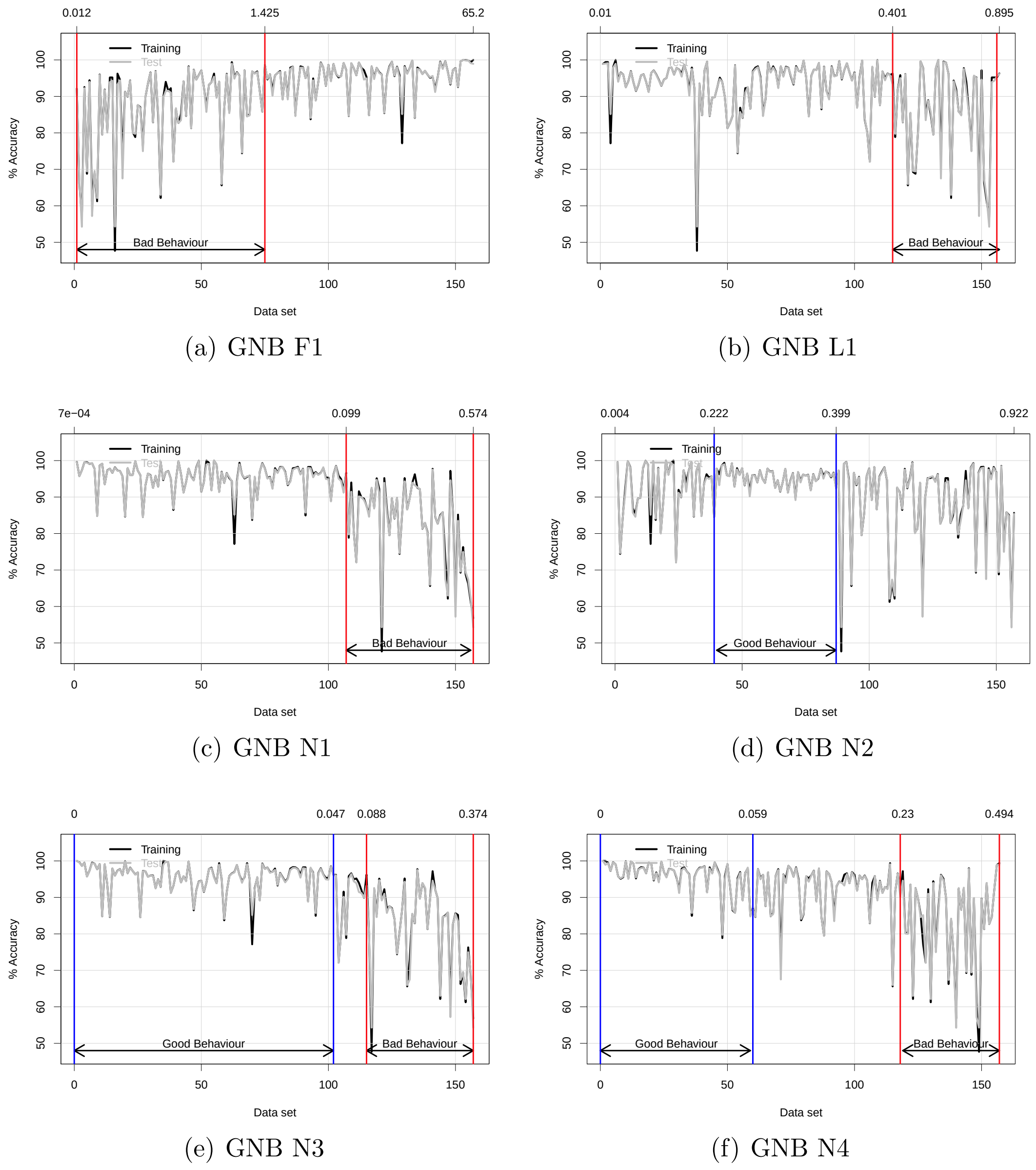


Figure 8.8: Datasets in increasing order of accuracy in training for Gaussian NB.

Here overfitting plays a much less important role, in fact, it is practically non-existent. However, the datasets could be more difficult to characterize since the patterns in the different graphs (Figure 8.9) are not as clear as for the previous studies on discrete domains. In this case, the CMs selected are different also: F1, L1, N1, N2, N3 and N4 provide the more meaningful patterns. Note that all the CMs related to the nearest neighbour paradigm seem to be the most informative.

The same parameters have been considered here except for *inTrainDifference* $= 3$ and *inTestDifference* $= 4$, since the range of the accuracy results obtained is different and the standard deviations is higher than in the



(a) GNB F1 (b) GNB L1

(c) GNB N1 (d) GNB N2

(e) GNB N3 (f) GNB N4

Figure 8.9: Characterization of the group of datasets into good or bad behaviour for Gaussian NB according to increasing order of 6 CMs individually: F1, N1, N4, L1, N3 and N2.

previous cases.

Similarly to the discrete domains, Tables 8.8 and 8.9 display the statistics of the good and bad intervals created from the 6 CMs selected and the disjunction and intersection rules for NB with Gaussians respectively.

Table 8.8: Rules for Gaussian NB from the intervals automatically obtained.

	Training accuracy=91.3758 ± 9.5240			Test accuracy=91.2430 ± 9.4023		
Good behaviour rules						
Id.	Interval	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
R1+	$N2 \in [0.222, 0.399]$	31.2101	95.5502	4.1744	95.3890	4.1459
R2+	$N3 \in [0.0, 0.047]$	65.6051	95.5554	4.1796	95.5673	4.3243
R3+	$N4 \in [0.0, 0.059]$	38.2166	95.8269	4.4511	95.7538	4.5108
Bad behaviour rules						
Id.	Interval	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
R1-	$F1 \in [0.012, 1.425]$	47.7707	86.9202	−4.4556	86.6906	−4.5524
R2-	$L1 \in [0.401, 0.895]$	26.7516	85.8407	−5.5351	85.3725	−5.8706
R3-	$N1 \in [0.099, 0.574]$	31.8471	82.4884	−8.8874	82.0932	−9.1498
R4-	$N3 \in [0.088, 0.374]$	26.7516	81.5791	−9.7967	81.2867	−9.9563
R5-	$N4 \in [0.23, 0.494]$	24.2038	82.5268	−8.8490	82.6842	−8.5589

Table 8.9: Disjunction and intersection rules from all simple rules for Gaussian NB.

Id.	If	Then (behaviour)	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
PRD	R1+ or R2+ or R3+	good	71.9745	95.0517	3.6759	95.0079	3.7649
NRD	R1- or R2- or R3- or R4- or R5-	bad	67.5159	88.9608	-2.4150	88.8034	-2.4396
PRD ∧ NRD	PRD and NRD	good	40.1274	93.9624	2.5866	93.9413	2.69832
PRD ∧ ¬NRD	PRD and ¬NRD	good	31.8471	96.4243	5.0485	96.3518	5.1088
NRD ∧ ¬PRD	NRD and ¬PRD	bad	27.3885	81.6330	-9.7429	81.2757	-9.9673
¬NRD ∧ ¬PRD	¬(PRD or NRD)	good	0.6369	94.9405	3.5647	94.4048	3.1618

Figure 8.10 shows the final characterization, clearly not as accurate as the previous ones for NB and AODE in discrete domains. Still, average accuracy on training is equal to 95.0508 ± 4.45 for good behaviour and 81.633 ± 12.24 for bad behaviour; whereas average accuracy on test is equal to 95.0026±4.16 for good behaviour and 81.2757 ± 11.9 for bad behaviour.

8.2.2.2 GAODE

Figure 8.11 shows the differences in terms of accuracy on training and test for the group of datasets ordered in ascending accuracy in training for GAODE.

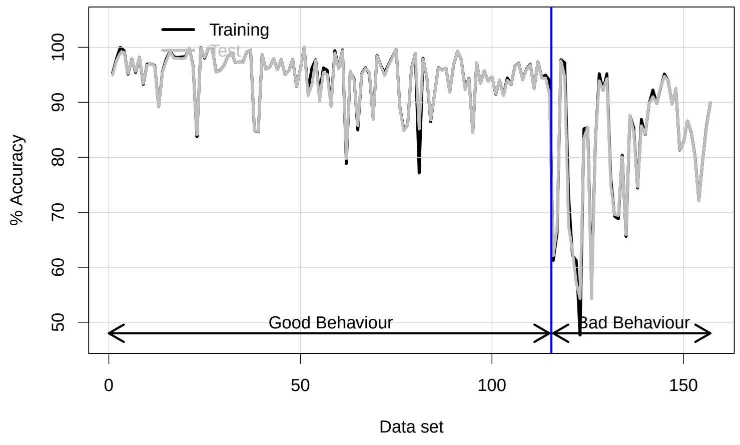


Figure 8.10: Gaussian NB characterization in terms of good and bad behaviour from the disjunction and intersection rules of 6 CMs: F1, L1, N1, N2, N3 and N4.

Also in this case, the use of Gaussians does not result in high overfitting.

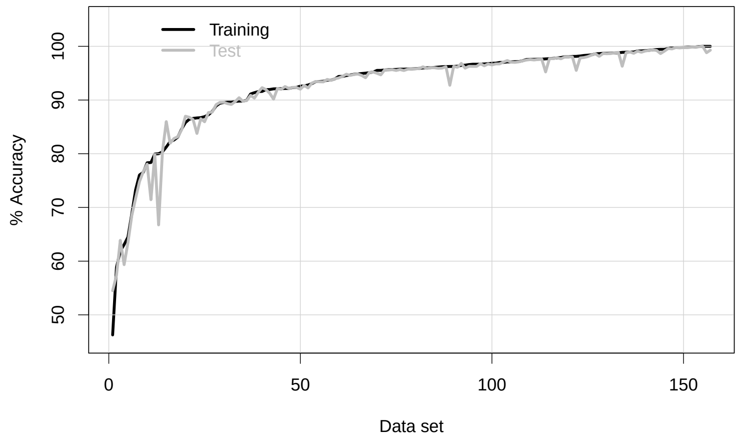


Figure 8.11: Datasets in increasing order of accuracy in training for GAODE.

The same group of CMs than for GNB has been selected, since they seem to provide the clearest patterns also for GAODE. However, the decisions on where to fix the boundaries of the intervals are harder to make now. Note, for example, Graph 8.12 (d), corresponding to N2, where the boundaries could arguably be placed differently.

Also, the same parameters have been considered here except for $inTrain\text{-}Difference = 3$ and $inTestDifference = 3.5$.

Once again, Tables 8.10 and 8.11 display the statistics of the good and bad intervals created from the 6 CMs selected and the disjunction and intersection rules for GAODE respectively. Despite the difficulties, we can observe a significant support for the different intervals.

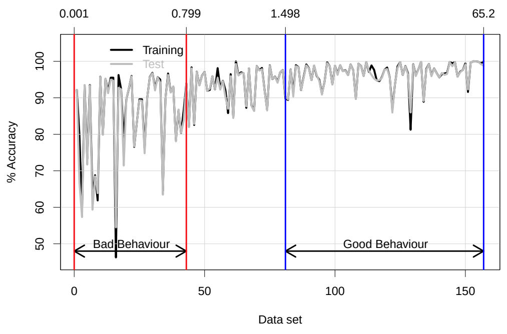
Table 8.10: Rules for GAODE from the intervals automatically obtained.

	Training accuracy=92.8283 ± 8.6279				Test accuracy=92.5401 ± 8.7134	
Good behaviour rules						
Id.	Interval	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
R1+	$F1 \in [1.498, 65.2]$	48.4076	96.6625	3.8342	96.5529	4.0128
R2+	$N1 \in [0.0007, 0.009]$	26.7516	96.6642	3.8359	96.6478	4.1077
R3+	$N3 \in [0.0, 0.0040]$	31.2101	96.6232	3.7949	96.5828	4.0428
R4+	$N4 \in [0.0, 0.038]$	29.9363	97.8349	5.0066	97.7090	5.1689
Bad behaviour rules						
Id.	Interval	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
R1-	$F1 \in [0.0010, 0.799]$	280255	85.3288	−7.4995	84.6278	−7.9123
R2-	$L1 \in [0.501, 0.895]$	22.2930	88.2348	−4.5936	87.0917	−5.4484
R3-	$N1 \in [0.112, 0.574]$	30.5732	85.07691	−7.7514	84.3415	−8.1986
R4-	$N2 \in [0.46, 0.922]$	31.8471	89.0950	−3.7333	88.2554	−4.2846
R5-	$N3 \in [0.049, 0.374]$	34.3949	86.1128	−6.7155	85.4249	−7.1151
R6-	$N4 \in [0.234, 0.494]$	23.5669	84.3965	−8.4318	84.2006	−8.3394

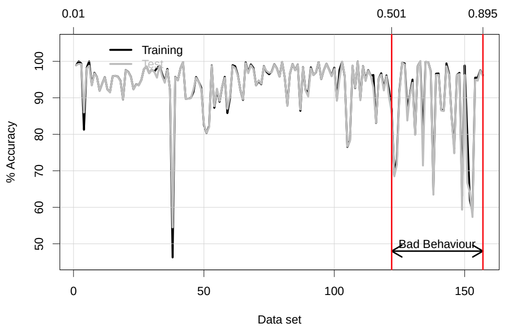
Table 8.11: Disjunction and intersection rules from all simple rules for GAODE.

Id.	If	Then (behaviour)	%Support	%Training Accuracy	Training Difference	%Test Accuracy	Test Difference
PRD	R1+ or R2+ or R3+ or R4+	good	60.5095	96.2804	3.4521	96.1927	3.6526
NRD	R1- or R2- or R3- or R4- or R5- or R6-	bad	62.4204	90.5509	-2.2774	90.1524	-2.3876
PRD ∧ NRD	PRD and NRD	good	27.3885	95.5664	2.7380	95.4612	2.9211
PRD ∧ ¬NRD	PRD and ¬NRD	good	33.1210	96.8708	4.0425	96.79764	4.2576
NRD ∧ ¬PRD	NRD and ¬PRD	bad	35.0318	86.6297	-6.1986	86.0019	-6.5381
¬NRD ∧ ¬PRD	¬(PRD or NRD)	good	4.4586	94.6822	1.8539	94.3393	1.7993

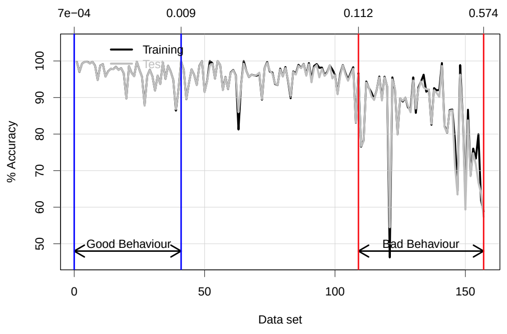
Figure 8.13 displays the final characterization for GAODE. Average accuracy on training is equal to 96.1707±3.48 for good behaviour and 86.6297±



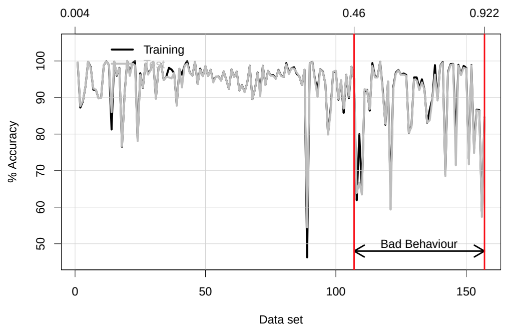
(a) GAODE F1



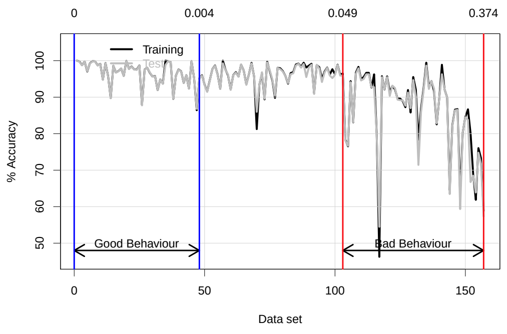
(b) GAODE L1



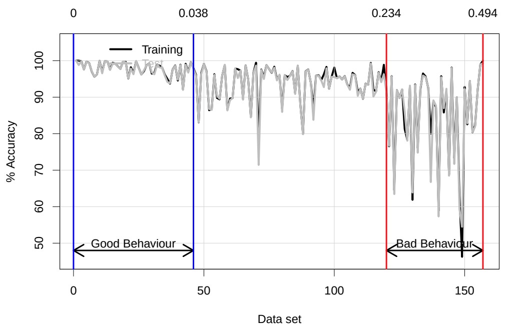
(c) GAODE N1



(d) GAODE N2



(e) GAODE N3



(f) GAODE N4

Figure 8.12: Characterization of the group of datasets into good or bad behaviour for GAODE according to increasing order of 6 CMs individually: F1, L1, N1, N2, N3 and N4.

11.49 for bad behaviour; whereas average accuracy on test is equal to 96.0655 ± 3.25 for good behaviour and 86.0019 ± 8.69 for bad behaviour.

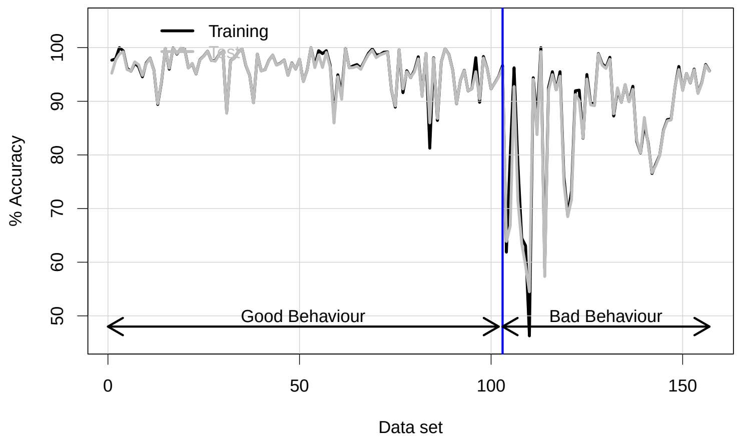


Figure 8.13: GAODE characterization in terms of good and bad behaviour from the disjunction and intersection rules of 6 CMs: F1, L1, N1, N2, N3 and N4.

8.3 Change in behaviour on CMs when discretizing the datasets

Since most of the BNCs work in the discrete domain, we wonder how much the values of these measures change when they are computed directly on a continuous dataset and its discretized version.

When a numeric dataset is discretized, independently of the discretization method used, the calculations for the measures designed to characterize the apparent complexity of datasets for supervised learning change as well.

It is possible that depending on the dataset, the discretized version becomes easier or more complex to classify. It is even more likely that for a particular discretized version of a dataset, some measures indicate that it is now easier to classify, whereas according to other measures, the problem becomes more complex.

Measure	Simpler if...	Dif. Disc.	Tendency	Simpler
F1v	+	4.2077	+	Yes
F1	+	-2.7438	−	No
L1	−	0.7021	+	No
F2	−	0.3341	+	No
F4	+	-0.2549	−	No
L3	−	-0.2318	−	Yes
N2	−	0.1964	+	No
F3	+	-0.0829	−	No
T1	−	0.0597	+	No
L2	−	-0.0592	−	Yes
N4	−	0.0371	+	No
N3	−	0.0028	+	No
N1	−	0.0013	+	No
T2	+	0	=	Same

Table 8.12: Changes in behaviour observed on the CMs when calculated on numeric datasets and their discretized versions. The measures are organised in decreasing order according to the absolute value of the variability observed.

We believe that, for most of the measures, the tendency in variation is similar for all the datasets, and hence, we would like to empirically know what this tendency is.

In order to study how the different measures in Table 8.1 change, we calculate the values for these measures on the original numeric datasets and also on their discretized versions. When the values obtained are compared and the patterns analysed, we obtain the results included in Table 8.12.

In the third column in Table 8.12, the average differences between the values for a measure in the original dataset and the discretized version are displayed. The general pattern of the differences between the discretized version and the original one is shown in the fourth column by: a symbol +, if a particular measure increases in the discretized version; −, if it decreases; or =, if it remains equal. The fifth column indicates whether the change in a particular measure implies that the problem becomes more or less simple after discretization. The main results summarize as follows:

- F1 and F1v, which deal with the discriminative ratio of attributes, seem to be the most affected. Note that whereas the former increases, the latter decreases, thus reflecting simpler datasets to classify with respect to F1 and the opposite according to F1v.

- For F2, the original numeric datasets have values very close to 0. Those that are 0 remain in the discrete version, but others increase their value indicating, at least in theory, a more complex dataset to classify.
- From F3 (double line) and going downwards in Table 8.12, we can find the most stable measures (with differences lower than ± 0.1). F3 decreases in most of the cases, although not much. L2 is another quite stable measure, that decreases a bit, indicating a simpler problem for linear classification. N1 is the most stable, followed by N3 and N4. T1 slightly increases whereas T2 remains the same, as expected given its definition.

To summarize, even though a common pattern is observed for each measure on the group of datasets, different changes are observed for different measures with respect to how easy or difficult the classification of a particular dataset becomes after discretization. Even so, we believe that these results might vary depending on the group of datasets considered, or even the discretization method. Hence, they should be carefully taken into account.

8.4 Meta-classification of semi-naive BNCs

As a different and more practical approach into the study of the differences between the domains of competence for some semi-naive BNCs, we propose a mechanism to select the most promising classifier (belonging to this family) for a particular dataset, based on the values of some of the CMs.

There have been several studies oriented to compare classifiers of different nature. One of the most ambitious and rigorous is the *Statlog project* [King *et al.*, 1995; Michie *et al.*, 1994], where about 20 procedures are compared for about 20 real datasets, mainly focused on error rates. Besides, a meta-level machine learning rule for algorithm recommendation, called the *Application Assistant* [Brazdil *et al.*, 1994], that uses the C4.5 algorithm [Quinlan, 1993] to construct the rules from the given data is proposed. These rules are based on the number of instances, attributes and classes; the proportion of binary, categorical or unknown attributes; the use of cost or the value of other statistical measures. However, the rules generated by the expert system were not very meaningful, mainly due to a lack of training data [van der Walt & Barnard, 2006]. In Sohn [1999] a ranking of classification algorithms is provided by using a statistical meta-model. It is used to predict the expected classification performance of each algorithm as a function of several data characteristics.

We pretend further automatize this process by using the values of the CMs for supervised classification considered in previous sections. The idea is that, given a particular dataset to be classified, it is possible to predict which semi-naive BNC is more likely to provide the most accurate predictions. For this purpose, it is necessary to create what we call a *training meta-dataset*: where every instance represents a single dataset, for which the predictive attributes correspond, in principle, to the 14 complexity measures in Table 8.1. This is in concordance with the method proposed in Hernández-Reyes *et al.* [2005]. However, the biggest difference between their approach and ours is that they consider a single class label dataset, assigning to every instance the classifier with the lowest error for the dataset that represents that instance, i.e. the classifier that obtains the lowest rate among KNN, NB and C4.5.

In our case, only semi-naive BNCs for discrete attributes are considered, in particular: NB, AODE, HODE, TAN and KDB3[1]. We believe that the selection of a single classifier based directly on the lowest error value can be too arbitrary. Alternatively, we propose to carry out statistical tests on the classifiers' results for each problem, in order to keep the best classifier and also those whose error rates are not significantly worse. Given that the considered classifiers belong to the same family, it is reasonable to expect small differences.

This then requires to resort to multi-label classification [Tsoumakas & Katakis, 2007] in order to handle the existence of multiple class labels[2]. Since,

[1]KDB with $k = 3$ has been selected for these experiments in order to gain some variety among the classifiers considered.

[2]Also known as multi-dimensional classification [Bielza *et al.*, 2011]. Note, in any case, the difference with a multi-class problem, that simply refers to the existence of one class with more than two labels.

once again, 5x2cv is used for the evaluation process, the 5x2cv F Test defined by Alpaydin [1999] has been used to select the semi-naive BNCs for each dataset. The level of significance has been fixed at 95% ($\alpha = 0.05$).

In order to construct the meta-dataset with the complexity measure values from different datasets, we select the original group of 26 numeric datasets in Table 4.1 (Section 4.4). We discretize them applying EF5 as well.

A small sample of this training meta-dataset can be found in Table 8.13. Every example corresponds to the result of the 14 complexity measures for a specific dataset, whereas the class labels are binary and correspond to the 5 following semi-naive BNCs: NB, AODE, HODE, TAN and KDB3: a bit equal to 1 for the j^{th} class label on the i^{th} instance indicates that the classifier on j, either obtain the best error rate for the dataset on the i^{th} position or it is not significantly worse that the classifier that does.

Table 8.13: Sample of the meta-dataset created to predict the best semi-naive BNC based on data complexity measures.

dataset	F1	F1v	F2	F3	F4	L1	L2	L3	N1	N2	N3	N4	T1	T2	NB	AODE	HODE	TAN	KDB3
a1iris.2c0	0.474	16.455	0.000	0.807	1.000	0.333	0.007	0.000	0.067	0.209	0.013	0.263	1	37.5	0	1	0	0	0
a1iris.2c1	0.458	2.374	0.250	0.433	0.573	0.658	0.333	0.500	0.200	0.264	0.093	0.250	1	37.5	0	0	1	0	0
a1iris.2c2	0.475	10.186	0.062	0.627	0.760	0.461	0.053	0.017	0.160	0.244	0.093	0.290	1	37.5	1	1	1	0	1
⋮			⋮			⋮			⋮			⋮			⋮				⋮

The statistics of the resulting meta-dataset are summarized in Table 8.14. The upper part of the table (above the horizontal line) displays general information: such as the number of examples, attributes or class labels; whereas the lower part includes more specific information related to the class labels. The number of distinct labelsets indicates the binary combinations out of the 2^5 possible. The value for cardinality is calculated as the ratio of positive bits (those equal to 1) in the labels over the total number of instances. The density is just the division of the cardinality between the number of labels. This values indicate that, on average, every example can be optimally classified by at least 2 semi-naive BNCs. Two figures to be highlighted here are: the number of examples of cardinality equal to 1, i.e., those that only have one "best" classifier, which is 39; and the number of trivial examples (cardinality equal to 5), i.e., those for which any classifier would be equally

Examples:	157	**Labels:**	5 (binary)
Predictive attributes:	14 (numeric)		
Distinct Labelsets:	24		
Cardinality:	2.52	**Density:**	0.50
***Percentage of examples with label:**			
1(NB):	19.74%	4(TAN):	52.23%
2(AODE):	56.59%	5(KDB3):	61.78%
3(HODE):	61.15%		
***Examples of cardinality:**			
0:	0	3:	43
1:	39	4:	19
2:	43	5:	13

Table 8.14: Statistics of the meta-dataset created.

eligible, which is 13.

Once the meta-dataset is ready, we can simply use a multi-label classifier to handle it. Several strategies exist for multi-label classification; some of them transform the multi-label classification problem into one or more single-label classification problems, and others simply extend specific learning algorithms in order to handle multi-label data directly. For our experiments, we select the following two approaches to carry out the meta-classification task:

- **Binary relevance** (BR): is a transformation method that learns a binary classifier for each class label. In our case, we transform the original dataset into 5 binary datasets that contain all the examples of the original one, labelled positively for datasets i if the label set of the original example contained label i, and negatively otherwise. For the classification of a new instance, the original definition of BR would output the union of the labels that are positively predicted by the 5 classifiers. In our case, only the most likely label is returned.

- **RA*k*EL**: [Tsoumakas & Vlahavas, 2007], is a random k-labelset method that constructs an ensemble of label powerset (LP) classifiers, where each LP is trained using a different small random subset of the set labels. LP is a simple but effective problem transformation method that works as follows: it considers each unique set of labels that exists in a multi-label training set as one of the classes of a new single-label classification task. A ranking of the labels is produced by averaging the zero-one predictions of each model per considered label. Thresholding is used to produce a bipartition as well. Only the first label in the ranking will be considered in our case.

However, all of these strategies consider a multi-label prediction phase as well. For our purposes, even though we are training a classifier based on a multi-label paradigm, we select only the best classifier to predict with. This restriction requires to redefine the way in which the evaluation is performed, so that a specific prediction, Z_e for the example (e, Y_e), is considered successful if the label predicted is among those included in Y_e, where both Z_e and Y_e are binary vectors of length L, L being the number of semi-naive BNCs considered, i.e., the number of labels. However, given that the number of positive values in Z_e is only one, we can use the original definition of **example-based precision** [Tsoumakas *et al.*, 2010] as evaluation measure:

$$Precision = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Z_i|}, \tag{8.1}$$

where the operator $|.|$ indicates the cardinality of the positive bits. Consider, for example, an output $Y_e = \{0, 1, 1, 0, 0\}$, which indicates that both AODE and HODE provide the best results for a particular dataset e, i.e., one of them has the highest absolute value and the other is not significantly worse. Then, if the meta-classifier, let say NB with BR (NB-BR), provides the output $Z_e = \{0, 1, 0, 0, 0\}$, this example would contribute to the summation as 1. If the output provided by RA***k***EL were $Z_e = \{0, 0, 0, 0, 1\}$ instead, the contribution would be equal to 0.

Note that since the average number of "valid" labels for every instance is equal to 2.5, our classification problem can be considered to be of equivalent difficulty to a binary class problem, since there is a 50% of probability to be accurate when classifying.

Additionally, it may not be necessary to use all the complexity measures as predictive attributes, as some of them can be redundant, irrelevant and maybe the "intrinsic" dimensionality may be smaller than the total number of measures considered. To that aim, we find a large amount of literature for feature selection (FS) [Dash & Liu, 1997; Guyon & Elisseeff, 2003]. Note that FS is not required here for dimensionality reduction with efficiency purposes, but it could be beneficial to remove measures that are too similar in the meta-dataset, and hence, redundant.

The whole process is outlined in Figure 8.14. The left-hand side displays the three steps involved in the meta-dataset's formation, which entails the most time consuming part of the process. The steps required when a new dataset faces a classification process, is included in the dashed line. Given a new dataset, the values for the CMs considered in the meta-classification process will be calculated (not necessarily all of them, as shown in Section 8.4.1). From these values the meta-classifier selected will return the best semi-naive BNC.

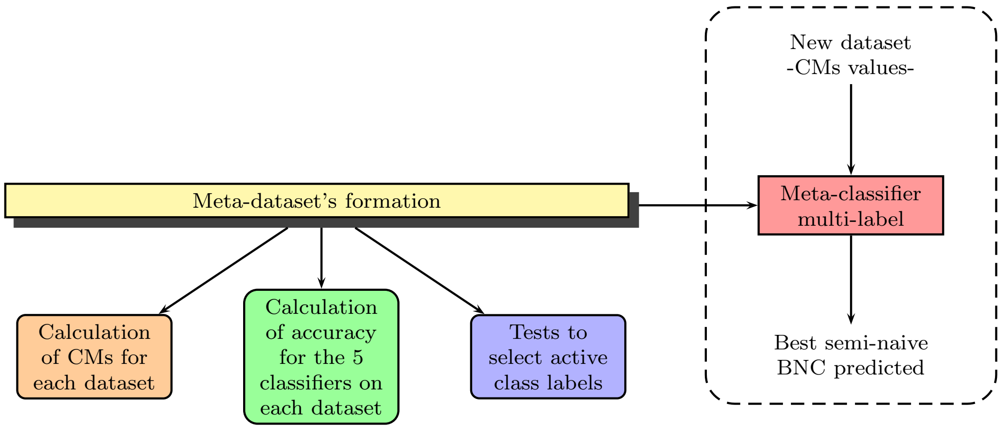

Meta-dataset's formation
Calculation of CMs for each dataset
Calculation of accuracy for the 5 classifiers on each dataset
Tests to select active class labels
New dataset -CMs values-
Meta-classifier multi-label
Best semi-naive BNC predicted

Figure 8.14: Schema of the meta-classification process.

8.4.1 Experimental methodology and results

We have resorted to a Java library for multi-label learning, called Mulan [Tsoumakas *et al.*, 2011], in order to handle the multiple labels. The two meta-classifiers selected to work with the meta-dataset created are NB with BR and RA***k***EL, described above.

The selection of these two multi-classifiers have been motivated by the results obtained with the different algorithms provided by Mulan. Other paradigms have been tested, such as a lazy learning approach (ML-KNN); and transformation methods, such as classifier chains [Read *et al.*, 2009] with different base classifiers. Even though this study is not an exhaustive one, since it does not cover all the multi-classifiers in the existing literature (such as Bielza *et al.* [2011]), we believe that it is sufficient for our purposes.

In Table 8.15, different results in terms of example-based precision are shown. The alternatives tested are as follows:

- The first column, `Data`, indicates whether the data considered are directly the value of the measures for the different datasets (Original) or the data have been transformed through principal component analysis techniques (PCA). Dimensionality reduction is accomplished by choosing enough eigenvectors to account for some percentage of the variance in the original data, which has been set to 0.95 (95%)[1] (PC space). Furthermore, the PC space data have been transformed back to the original space eliminating some of the worst eigenvectors, with the aim of filtering attribute noise (PC space transformed back to original space).

- Feature selection through clustering techniques has also been carried out in some cases, as indicated in the second column, `Clustering+FS`. More specifically, a k-means clustering algorithm is performed in the transposed dataset with[2] $k = 10$. The output indicates the following clusters: $(L1, N2)$, $(L2, L3, N3)$, $(F1, N1)$ and the rest of the measures in isolation. In order to select which measures to keep from each cluster we use PCA techniques again. The procedure is as follows: data are transformed through the PC space and back to the original space. As

[1]Default value in WEKA.

[2]Note that in this case, the purpose of feature selection is mainly carried out in order to remove possible noisy features, that is why we consider appropriate (although arbitrarily) not to remove more than 4 predictive attributes.

only the best PCs are retained, by setting the variance covered equal to 0.95, we will obtain a dataset in the original space but with less attribute noise as above. Hence, the ranking obtained by this method is:

$$F4, L2, L1, F1v, F1, F3, F2, N4, T2, T1, N1, L3, N3, N2$$

We maintain then: $L1$ from cluster $(L1, N2)$, $L2$ from $(L2, L3, N3)$ and $F1$ from $(F1, N1)$, along with the rest of the measures. And we will discard N2, L3, N3 and N1, which happen to be the last four attributes given by PCA.

- Two multi-label classifiers (BR-NB and RA*k*EL) are directly applied or after performing FS as explained above. Indicated in the third column, `Meta-Classifier`.

Table 8.15: Expected example-based precision for meta-classifier selection.

Data	Clustering+FS	Meta-Classifier	Precision ± Stand. dev.
Original		BR-NB	84.79±7.40
		RA*k*EL	86.75±7.59
	K-means+PCA	BR-NB	86.08±5.32
	K-means+PCA	RA*k*EL	86.04±7.30
PC space		BR-NB	85.46±10.4
		RA*k*EL	77.67±8.61
PC space transformed back to original space		BR-NB	86.08±6.63
		RA*k*EL	86.71±5.8
	K-means+PCA	BR-NB	86.08±6.01
	K-means+PCA	RA*k*EL	**87.38±7.81**

The results on the last column in Table 8.15, show a variety of accuracy values ranging from 77.7% to 87.4% depending on the data considered, the use or not of pre-processing techniques for FS and the multi-label classifier applied. It is obvious that the options and combinations here to test with are massive, and it is not our aim to perform an exhaustive study. The main purpose of this small comparison is to give an idea of the predictive power of the model.

All in all, the results seem to be encouraging, since in the best case, they offer a precision estimated in 87.38% of predicting correctly one of the best semi-naive BNCs, based on the complexity measures of a particular dataset with discrete attributes.

8.5 Conclusions and future work

This chapter provides a different view of the performance of some of the semi-naive BNCs considered in this dissertation. Motivated by the increasing evidence that an "almighty" classifier does not exist, we defined the characteristics of the datasets for which these semi-naive BNCs provide accurate results.

For this purpose, we have resorted to several complexity measures recently proposed, that have already shown their power for characterizing classifiers of different nature, although mainly on continuous datasets.

In this chapter we have characterized NB and AODE on discrete and continuous datasets. Contrary to our initial guess, it has been easier to do this on the discrete domains. We have tried to understand how the complexity measures' values change on continuous datasets and these same datasets after discretization. We can not assert that the values of these measures indicate simpler classification problems in general on the discrete datasets. However, the increase/decrease of difficulty in characterizing the new space formed by the values of the CMs on the discrete datasets, is independent of the lack of unanimity in the tendency followed by the changes. In fact, the space of the CMs values provided by the discrete datasets have been easier to characterize, as indicated above.

The most important result is that it is possible to characterize both NB and AODE for both domains and to obtain disjoint rules to predict if the classifier will perform well or poorly, depending on the values of some of the complexity measures.

To finalize, an automatic procedure to advise on the best semi-naive BNC to use for classification is proposed, with an estimated predictive accuracy of 87.38%.

The study carried out in this chapter can be easily extended on many points. Firstly, the test bed of considered datasets can be extended incorporating the datasets from the Landscape contest, that has been created to cover a wider range of the complexity measurement space. Secondly, in our work the measures' selection of the 14 initially considered has been made using empirical criteria, mainly based on the pattern provided by plotting the complexity measures. It may be the best method, it may be not. This remarks the need for a more theoretical way to know the reliability of a measure to characterize a particular classifier, which remains to be explored.

Thirdly, in Appendix A, a preliminary study on the relationships between pairs of measures to characterize accuracy results for NB has been included. It remains for future works to extend this promising study for other classifiers and, probably, for higher dimensionality relations.

Fourthly, we propose to consider the bias/variance decomposition in future studies. We have already introduced the limitations on the use of the accuracy for evaluation. Even though it provides a good general view of the performance, the use of bias and variance components in isolation could provide more knowledge about the good or bad performance of the different classifiers in several datasets.

Part V

Concluding remarks

Chapter 9

Conclusions and future work

Uncertainty and mystery are energies of life. Don't let them scare you unduly, for they keep boredom at bay and spark creativity.
R. I. Fitzhenry.

9.1 Conclusions

This dissertation is a contribution to the state of the art of semi-naive Bayesian network classifiers, focused on the *aggregating one-dependence estimators* paradigm [Webb *et al.*, 2005]. This contribution includes studies on different aspects of the semi-naive BNCs, such as new proposals to overcome AODE's limitations, the behaviour of these classifiers with traditional and non-disjoint discretization techniques and the domains of competence of this family of classifiers.

Part II of the thesis is devoted to AODE's limitations. Chapter 3 presents a new classifier named HODE [Flores *et al.*, 2009b], that provides a linear order in classification time and a reduction in space complexity compared to AODE. HODE estimates a new variable whose main objective is to model the significant dependencies between each attribute and the rest of the attributes that AODE takes into account. To estimate the number of states of this new variable it resorts to the EM algorithm, which makes it slower in training time. However, HODE is subject to be easily parallelized, and it may be a good alternative for high dimensional datasets due to memory constrains.

Chapter 4 presents two approaches, GAODE and HAODE [Flores *et al.*, 2009a], to handle continuous attributes when applying the AODE paradigm. GAODE applies conditional Gaussian networks to model the relationships between each predictive attribute and its parents, obtaining competitive results compared to AODE. GAODE implies a reduction in the space complexity and its parameters can be computed *a priori* in a single pass over the data, maintaining AODE's time complexity as well. HAODE, in turn, keeps the superparent attribute discrete in every model. This approach offers the clear advantage of dealing with any kind of dataset. HAODE is generally competitive with AODE, and even better for datasets with continuous attributes and no missing data.

In Chapter 5 we propose the use of Mixture of Truncated Exponentials to generalize the application of AODE to all kind of datasets [Flores *et al.*, 2011b]. Even though it is a good alternative for some datasets in order to avoid Gaussian assumptions, the results indicate that the use of MTE estimations requires selecting the proper number of intervals, into which the domain of leaf variables in the mixed tree is split, in order to compete with discretization methods.

Part III is devoted to the study of the impact of several discretization paradigms on the family of semi-naive BNCs. Chapter 6 compares some of the most common discretization methods, whereas Chapter 7 investigates the use of non-disjoint intervals. The conclusions indicate that the ranking obtained with traditional disjoint techniques is the same for HAODE, AODE and BNHC, as their performance is sufficiently different; while the position in the ranking for NB, TAN and KDB, can vary in a particular case, since they obtain very similar results. Even so, in light of the results, we believe that if the set of datasets is large enough, the choice of discretization method is irrelevant when comparing the BN classifiers [Flores *et al.*, 2010, 2011a]. However, whereas some of the most common disjoint discretization techniques have failed to demonstrate consistent improvement relative to alternatives, non-disjoint discretization demonstrates better win/draw/loss records and significant overall improvement for AODE and HAODE [Martínez *et al.*, 2012].

Finally, in Part IV (Chapter 8), we try to find the parcels of good behaviour in the complexity measurement space for the family of semi-naive BNCs. Initially, patterns of good and bad behaviour are obtained to characterize both NB and AODE in discrete and continuous domains. Unexpectedly, since discrete domains had been barely explored, the characterization process has been easier in this case. In addition, an automatic procedure to advise on the best semi-naive BNCs to use for classification has been proposed, with a promising predictive accuracy.

9.2 Future work

Throughout the different chapters of this dissertation some new proposals and ideas specific to each topic have been expounded. The main ones will be summarized here along with some new.

As far as HODE is concerned, it would be of a major interest to investigate how the estimations on one step in the EM algorithm used in HODE can be reused on posterior steps [Karciauskas, 2005].

Regarding the study on new classifiers to overcome AODE's limitations, it would be attractive to further investigate the proper general configuration of MTEs to obtain more successful results. One way could be the study of a new supervised method to dynamically search for the optimum number in every case into every dataset, with the aim to find a good trade-off between fitting and generalization capability of the model.

An exhaustive study on model and attribute selection, through the use of different metrics, would be a good complement to what has been studied in this dissertation also. It not only would provide improvements in terms of accuracy results, but also in time and space complexities (since simpler models would be obtained).

With respect to the non-disjoint discretization proposal for AODE and HAODE, we find that the study of the performance of these classifiers on a test bed of very high dimensional datasets, would allow to gain insight into the proposal based on weighting importance.

Regarding the domains of competence of the semi-naive BCNs, it remains for future works to investigate higher dimensionality relations between complexity measures, which can be the key to find more accurate characterizations of these classifiers. Furthermore, the use of the datasets created for the Landscape contest, that have been specifically created to cover a much wider range of the complexity measurement space, remains to be tackled.

Finally for now, and further moving off the path followed in this thesis, we believe that there are several ideas related to other paradigms, such as multinets or recursive nets, that could benefit from AODE's spirit, offering interesting alternatives for classification. Additionally, the study of the extension of the AODE paradigm to multi-label classification, where an example may be associated with multiple labels. An interesting work with Bayesian networks has been recently done in relation to this [Bielza *et al.*, 2011], however, we believe that there is still room for improvement as far as the introduction of semi-naive Bayesian network classifiers is concerned for example. Similar reasoning with the topic of multi-instance learning, where the examples are represented by more than one feature vector.

Appendix A

Domains of competence: bivariate relationships between complexity measures

A.1 NB on discrete data

Figures A.1, A.2 and A.3 show a selection of pairs of complexity measures to reflect patterns on accuracy values. More specifically, the X and Y-axis indicate the ranges of the complexity measures (original or logarithmic scale values depending on the graph). The colour of the points reflect different accuracy values obtained by NB with 5x2cv on the same 157 discrete datasets used in Section 8.2. Yellow and orange colours indicates accuracy values below the average (92.3230 ± 6.8575), more precisely, all the points below the average minus the standard deviation are yellow and the rest in orange. On the contrary, accuracy values over the average are displayed in blue and black, where black represents the top best accuracy values.

Figure A.1 displays those pairs of complexity measures where the behaviour in terms of accuracy for NB seems to depend mainly on only one

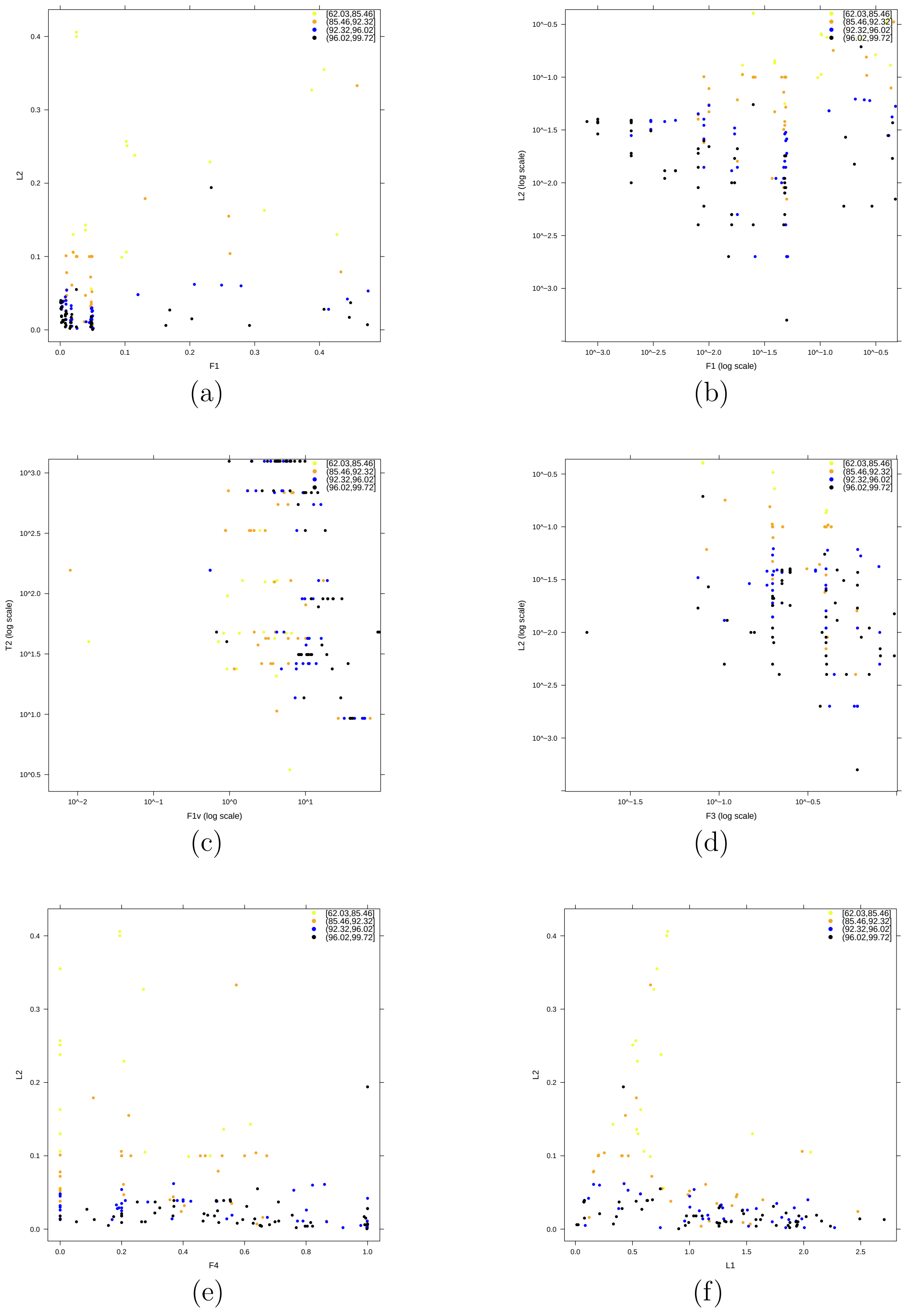


Figure A.1: Bivariate relationships for NB - EF5. Behaviour seems to depend mostly on the values of a single complexity measure.

of the variables. Subfigures A.1 (a) and (b) represent the same data with original and logarithmic scale respectively. One or the other scale will be displayed indistinctly depending of which one provides more illustrative results.

Subfigure A.2 (a) shows the positive correlation between the complexity measures N1 and N3. Subfigure A.2 (b), in turn, shows the relationship between T2 and L2, where large values of the latter indicates bad behaviour (as expected), but more interesting is the fact that very large values of T2 (average number of points per dimension) provides very good accuracy results (black points in all cases).

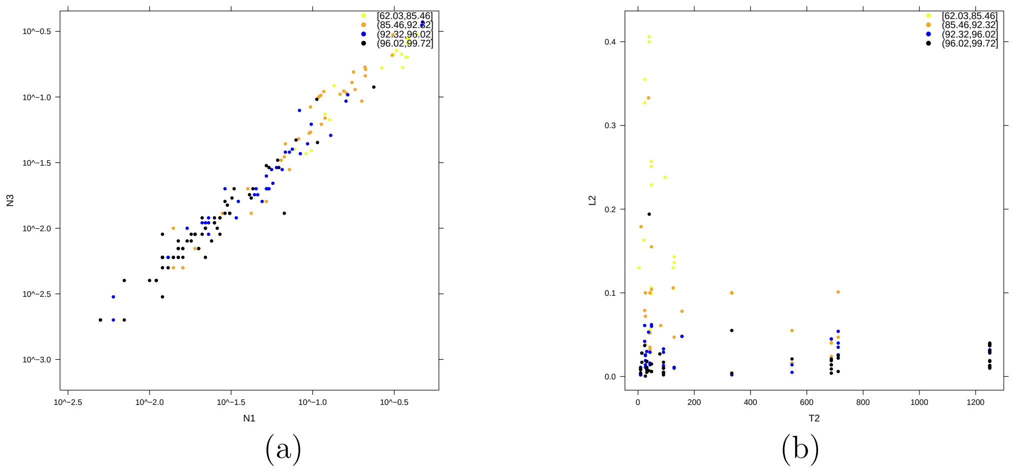
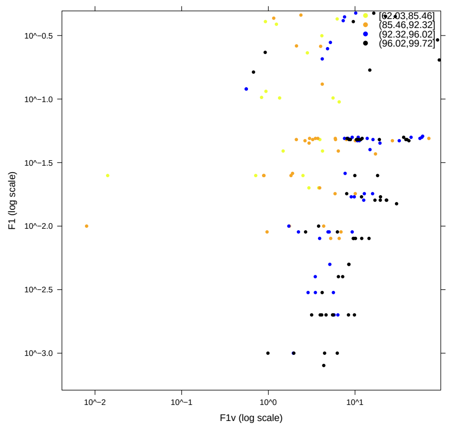


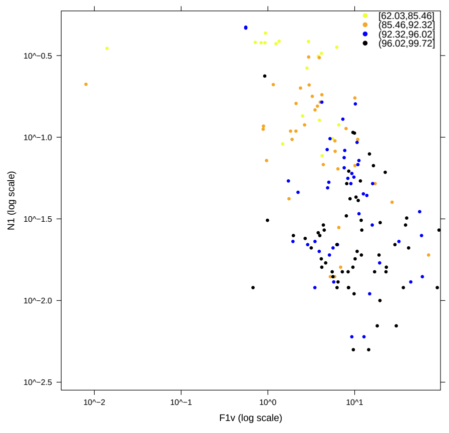
Figure A.2: Bivariate relationships for NB - EF5. Interesting relationships.

On the other hand, Figure A.3 displays bivariate relationships where the behaviour in terms of accuracy seems to depend in a pair of measures.

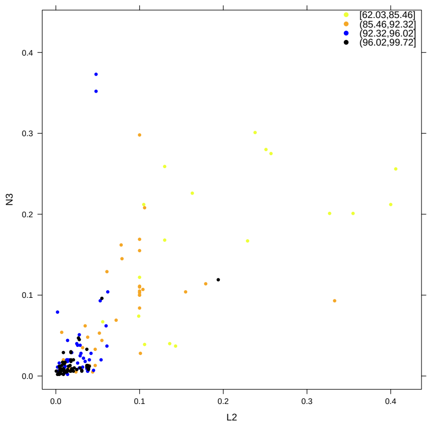
All in all, this only pretends to be an *apéritive*, to show how several measures in isolation seem to be sufficient to characterize, in this case, NB (Figure A.1); whereas others have more power jointly with at least one other measure (Figure A.3).



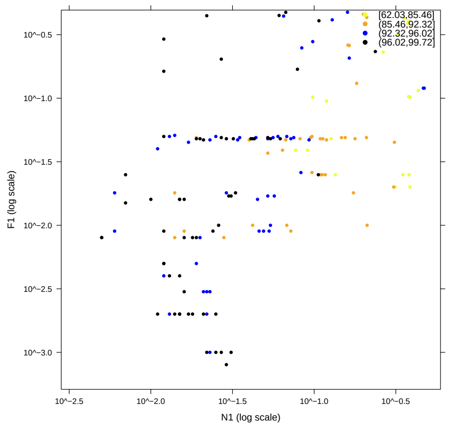
(a)



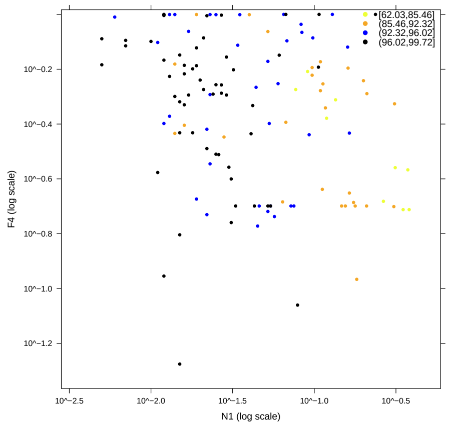
(b)



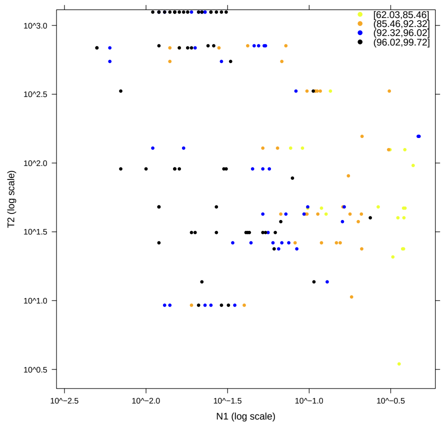
(c)



(d)



(e)



(f)

Figure A.3: Bivariate relationships for NB - EF5. Behaviour seems to depend on the joint values of the two complexity measures.

Appendix B

Publications

Part of the contents presented in this dissertation are the results of the following publications:

1. Flores, M.J., Gámez, J.A. & Martínez, A.M. (2012). *Intelligent Data Analysis for Real-Life Applications: Theory and Practice*, chap. Supervised Classification with Bayesian Networks: A Review on Models and Applications. Book chapter. Announced to be published on May 2012.

2. Martínez, A.M., Webb, G.I., Flores, M.J. & Gámez, J.A. (2012). Non-disjoint discretization for aggregating one-dependence estimator classifiers. In *Hybrid Artificial Intelligent Systems - 7th International Conference, Part II (HAIS 2012)*, vol. 7209 of *Lecture Notes in Computer Science*, 151–162.

3. Flores, M.J., Gámez, J.A., Martínez, A.M. & Salmerón, A. (2011). Mixture of Truncated Exponentials in Supervised Classification: case study for Naive Bayes and Averaged One-Dependence Estimators. In *11th International Conference on Intelligent Systems Design and Applications (ISDA 2011)*, 593–598.

4. Flores, M.J., Gámez, J.A., Martínez, A.M. & Puerta, J.M. (2011). Handling Numeric Attributes when comparing Bayesian Net-

work Classifiers: does the discretization method matter? *Applied Intelligence*, **34**, 372–385.

5. Flores, M.J., Gámez, J.A., Martínez, A.M. & Puerta, J.M. (2010). Deux Nouveaux Classifieurs basés sur AODE afin de traiter Variables Continues. In *Les 5èmes Journées Francophones sur les Réseaux Bayésiens (JFRB)*, Société Française de Statistiques (SFdS) - Association int. francophone d'EGC, http://hal.inria.fr/docs/00/46/68/59/PDF/GAODEetHAODE.pdf.

6. Flores, M.J., Gámez, J.A., Martínez, A.M. & Puerta, J.M. (2010). Analyzing the impact of the Discretization method when Comparing Bayesian Classifiers. In *The Twenty Third International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA-AIE 2010)*, 570–579.

7. Flores, M.J., Gámez, J.A., Martínez, A.M. & Puerta, J.M. (2009). GAODE and HAODE: two Proposals based on AODE to deal with Continuous Variables. In *Proceedings of the 26th Annual International Conference on Machine Learning* (ICML 2009), 313–320, ACM, New York, NY, USA.

8. Flores, M.J., Gámez, J.A., Martínez, A.M. & Puerta, J.M. (2009). HODE: Hidden One-Dependence Estimator. In the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*(ECSQARU 2009)*, vol. 5590 of *Lecture Notes in Computer Science*, 481–492.

9. Flores, M.J., Gámez, J.A., Martínez, A.M. & Puerta, J.M. (2009). Estudio y Comparativa de diferentes Discretizaciones en Clasificadores Bayesianos. In *Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2009)*, 265–274.

Other publications whose contents are not included in this dissertation are:

10. Bermejo, P., Gámez, J.A., Martínez, A.M. & Puerta, J.M. (2012). Algoritmos de estimación de distribuciones para la selección simultánea de instancias y atributos. In *VIII Congreso español sobre metaheurísticas, algoritmos evolutivos y bioinspirados (MAEB 2012)*, 31–38.

11. Zhong, S., Martínez, A.M., Nielsen, T.D. & Langseth, H. (2010). Towards a more Expressive Model for Dynamic Classification. In the 23rd Florida Artificial Intelligence Research Society Conference *(FLAIRS 2010)*, AAAI Press.

References

Aguilera, P.A., Fernández, A., Reche, F. & Rumí, R. (2010). Hybrid Bayesian network classifiers: Application to species distribution models. *Environmental Modelling & Software*, **25**, 1630 – 1639. 9

Akaike, H. (1978). A Bayesian analysis of the minimum AIC procedure. *Annals of the Institute of Statistical Mathematics*, **30**, 9–14. 53

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J. & García, S. (2011). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, **17**, 255–287. 37

Alpaydin, E. (1999). Combined 5 x 2 cv f test for comparing supervised classification learning algorithms. *Neural Computation*, **11**, 1885–1892. 71, 146

Andersen, S.K., Olesen, K.G., Jensen, F.V. & Jensen, F. (1989). HUGIN–A shell for building Bayesian belief universes for expert systems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 1080–1085. 26

Antal, P., Fannes, G., Timmerman, D., Moreau, Y. & Moor, B.D. (2003). Bayesian applications of belief networks and multilayer perceptrons for ovarian tumor classification with rejection. *Artificial Intelligence in Medicine*, **29**, 39–60. 9

Armañanzas, R., Inza, I.n. & Larrañaga, P. (2008). Detecting reliable gene interactions by a hierarchy of Bayesian network classifiers. *Computer Methods and Programs in Biomedicine*, **91**, 110–121. 9

Armañanzas, R. (2009). *Consensus policies to solve bioinformatic problems through Bayesian network classifiers and estimation of distribution algorithms*. Ph.D. thesis, Departamento de Ciencias de la Computación e Inteligencia Artificial. Universidad del País Vasco. 8

Axelson, D., Standal, I., Martinez, I. & Aursand, M. (2009). Classification of wild and farmed salmon using Bayesian belief networks and gas chromatography-derived fatty acid distributions. *Journal of Agricultural and Food Chemistry*. 9

Azzalini, A. & Bowman, A.W. (1990). A look at some data on the old faithful geyser. *Applied Statistics*, **39**, 357–365. 34, 83

Baker, M., Carpenter, B., Fox, G., Ko, S.H. & Lim, S. (1999). mpiJava: an object-oriented java interface to mpi. In *Proceedings of the International Workshop on Java for Parallel and Distributed Computing, IPPS/SPDP*. 59

Bernadó-Mansilla, E. & Ho, T.K. (2004). On classifier domains of competence. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04) Volume 1 - Volume 01*, 136–139. 37, 121

Bernadó-Mansilla, E. & Ho, T.K. (2005). Domain of competence of xcs classifier system in complexity measurement space. *IEEE Transactions on Evolutionary Computation*, **9**, 82–104. 37, 121

Bernard (1986). *Density Estimation for Statistics and Data Analysis (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. Chapman and Hall/CRC, 1st edn. 29

Bielza, C., Li, G. & Larrañaga, P. (2011). Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, **52**, 705–727. 145, 150, 160

BOUCKAERT, R.R. (2005). Bayesian network classifiers in WEKA. Tech. rep. 100

BRAZDIL, P., GAMA, J.A. & HENERY, B. (1994). Characterizing the applicability of classification algorithms using meta-level learning. In *Proceedings of the European conference on Machine Learning*, 83–102, Springer-Verlag New York, Inc., Secaucus, NJ, USA. 144

BRESSAN, G.M., OLIVEIRA, V.A., HRUSCHKA, E.R., JR. & NICOLETTI, M.C. (2009). Using Bayesian networks with rule extraction to infer the risk of weed infestation in a corn-crop. *Engineering Applications of Artificial Intelligence*, **22**, 579–592. 9

BUNTINE, W. (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, **8**, 195–210. 90

CASELLA, G. & BERGER, R. (2001). *Statistical Inference*. Duxbury Resource Center. 102

CERQUIDES, J. & DE MÁNTARAS, R.L. (2005). Robust Bayesian linear classifier ensembles. In *Proceedings of the 16th European Conference on Machine Learning (ECML-05)*, 72–83. 14

CHEESEMAN, P. & STUTZ, J. (1996). Advances in knowledge discovery and data mining. chap. Bayesian classification (AutoClass): theory and results, 153–180, American Association for Artificial Intelligence, Menlo Park, CA, USA. 50

CHMIELEWSKI, M.R. & JERZY (1996). Global discretization of continuous attributes as preprocessing for machine learning. *International Journal of Approximate Reasoning*, **15**, 319–331. 20

CHOW, C. & LIU, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, **14**, 462–467. 14

Cobb, B., Shenoy, P. & Rumí, R. (2006). Approximating probability density functions with mixtures of truncated exponentials. *Statistics and Computing*, **16**, 293–308. 79

Dash, M. & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, **1**, 131–156. 149

Deaton, A. (1997). *The analysis of household surveys : a microeconometric approach to development policy*. Published for the World Bank [by] Johns Hopkins University Press, Baltimore, MD. 30

DeGroot, M.H. (1970). *Optimal Statistical Decisions*. McGraw-Hill, New York. 25

Dempster, A.P., Laird, N.M. & Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, **39**. 48, 50

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, **7**, 1–30. 71, 73, 92, 113

Dietterich, T.G. (1998). Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, **10**, 1895–1923. 71

Domingos, P. & Pazzani, M.J. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, **29**, 103–130. 111

Dougherty, J., Kohavi, R. & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, 194–202. 21

Duda, R.O. & Hart, P.E. (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc. 4, 12

Elvira-Consortium (2002). Elvira: An environment for creating and using probabilistic graphical models. In *Probabilistic Graphical Models*. 80

Fayyad, U.M. & Irani, K.B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1022–1027. 20, 22, 55, 71, 80, 111

Fernandes, J.A., Irigoien, X., Goikoetxea, N., Lozano, J.A., naki Inza, I., Pérez, A. & Bode, A. (2010). Fish recruitment prediction, using robust supervised classification methods. *Ecological Modelling*, **221**, 338 – 352. 9

Fernández, A. & Salmerón, A. (2008a). Bayeschess: A computer chess program based on Bayesian networks. *Pattern Recognition Letters*, **29**, 1154–1159. 8

Fernández, A. & Salmerón, A. (2008b). Extension of Bayesian network classifiers to regression problems. In *Proceedings of the 11th Ibero-American conference on AI: Advances in Artificial Intelligence*, IBERAMIA '08, 83–92, Springer-Verlag, Berlin, Heidelberg. 34

Fisher, R.A. (1959). *Statistical Methods and Scientific Inference*. Oliver and Boyd, Edinburgh, 2nd edn. 92

Flesch, I., Fernández, A. & Salmerón, A. (2007). Incremental supervised classification for the MTE distribution: a preliminary study. In I.R. Ruíz & H.P. Cintas, eds., *Actas de Simposio de Inteligencia Computacional, SICO'2007*, 217–224. 34

Flores, J.L., Inza, I. & Larra (2007). Wrapper discretization by means of estimation of distribution algorithms. *Intelligent Data Analysis*, **11**, 525–545. 20

Flores, M.J., Gámez, J.A., Martínez, A.M. & Puerta, J.M. (2009a). GAODE and HAODE: two proposals based on AODE to deal with continuous variables. In A.P. Danyluk, L. Bottou & M.L. Littman, eds., *ICML*, vol. 382 of *ACM International Conference Proceeding Series*, 40, ACM. 67, 68, 78, 90, 158

Flores, M.J., Gámez, J.A., Martínez, A.M. & Puerta, J.M. (2009b). Hode: Hidden One-Dependence Estimator. In *Proceedings of the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, ECSQARU '09, 481–492, Springer-Verlag, Berlin, Heidelberg. 49, 157

Flores, M.J., Gámez, J.A., Martínez, A.M. & Puerta, J.M. (2010). Analyzing the impact of the discretization method when comparing Bayesian classifiers. In *Proceedings of the 23rd international conference on Industrial engineering and other applications of applied intelligent systems - Volume Part I*, IEA/AIE'10, 570–579, Springer-Verlag, Berlin, Heidelberg. 91, 158

Flores, M.J., Gámez, J.A., Martínez, A.M. & Puerta, J.M. (2011a). Handling numeric attributes when comparing Bayesian network classifiers: does the discretization method matter? *Applied Intelligence*, **34**, 372–385. 91, 158

Flores, M.J., Gámez, J.A., Martínez, A.M. & Salmerón, A. (2011b). Mixture of truncated exponentials in supervised classification: Case study for the naive Bayes and averaged one-dependence estimators classifiers. In *11th International Conference on Intelligent Systems Design and Applications (ISDA), 2011*, 593 –598. 79, 158

Frank, A. & Asuncion, A. (2010). UCI machine learning repository. `http://archive.ics.uci.edu/ml`. 70, 91, 111

Friedman, N., Geiger, D. & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, **29**, 131–163. 14, 17, 90

Gámez, J.A., Mateo, J.L., Nielsen, T.D. & Puerta, J.M. (2008). Robust classification using mixtures of dependency networks. In *Proceedings of the fourth European workshop on Probabilistic Graphical Models (PGM08)*, 129–136. 18

García, S. & Herrera, F. (2009). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, **9**, 2677–2694. 73, 92, 95

Geiger, D. & Heckerman, D. (1994). Learning Gaussian networks. In *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence*, 235–243. 66

Giang, P.H. & Shenoy, P.P. (2011). A decision theory for partially consonant belief functions. *International Journal of Approximate Reasoning*, **52**, 375–394. 34

Gupta, M.R. & Chen, Y. (2011). Theory and use of the EM algorithm. *Foundations and Trends in Signal Processing*. 50

Guyon, I. & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, **3**, 1157–1182. 149

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. & Witten, I.H. (2009). The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, **11**, 10–18. 13, 21, 70, 92

Hernández-Reyes, E., Carrasco-Ochoa, J.A. & Trinidad, J.F.M. (2005). Classifier selection based on data complexity measures. In A. Sanfeliu & M. Lazo-Cortés, eds., *CIARP*, vol. 3773 of *Lecture Notes in Computer Science*, 586–592, Springer. 37, 145

Hettich, S. & Bay, S.D. (1999). The UCI KDD Archive. `http://kdd.ics.uci.edu`. 111

Ho, T.K. (2001). Data complexity analysis for classifier combination. In J. Kittler & F. Roli, eds., *Multiple Classifier Systems*, vol. 2096 of *Lecture Notes in Computer Science*, 53–67, Springer. 36

Ho, T.K. & Basu, M. (2000). Measuring the complexity of classification problems. In *Proccedings of the 15th International Conference on Pattern Recognition (ICPR)*, 2043–2047. 36

Ho, T.K. & Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**, 289–300. 5, 36, 38, 41, 121

Hoekstra, A. & Duin, R.P.W. (1996). On the nonlinearity of pattern classifiers. *International Conference on Pattern Recognition*, 271. 42

Hoeting, J.A., Madigan, D., Raftery, A.E. & Volinsky, C.T. (1999). Bayesian model averaging : A tutorial (with discussion). *Statistical Science*, **14**, 382–417. 14

Hoeting, J.A., Madigan, D., Raftery, A.E. & Volinsky, C.T. (2000). Bayesian model averaging : A tutorial (with discussion) - correction. *Statistical Science*, **15**, 193–195. 14

Hruschka-Jr., E.R., Hruschka, E.R. & Ebecken, N.F.F. (2005). Applying Bayesian networks for meteorological data mining. In A. Macintosh, R. Ellis & T. Allen, eds., *SGAI Conf.*, 122–133, Springer. 9

Hsu, C.N., Huang, H.J. & Wong, T.T. (2000). Why discretization works for naive Bayesian classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, 399–406, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 24, 103

Hsu, C.N., Huang, H.J. & Wong, T.T. (2003). Implications of the Dirichlet assumption for discretization of continuous variables in naive Bayesian classifiers. *Machine Learning*, **53**, 235–263. 103

Iman, R. & Davenport, J. (1980). Approximations of the critical region of the fbietkan statistic. *Communications in Statistics - Theory and Methods*, **9 (6)**, 571–595. 92

Jaeger, M. (2003). Probabilistic classifiers and the concepts they recognize. In T. Fawcett & N. Mishra, eds., *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, 266–273, AAAI Press. 36

Jiang, L. & Zhang, H. (2006). Weightily averaged one-dependence estimators. In *Proceedings of the 9th Pacific Rim international conference on Artificial intelligence*, PRICAI'06, 970–974, Springer-Verlag, Berlin, Heidelberg. 14

Jiang, L., Zhang, H., Cai, Z. & Su, J. (2005). Learning tree augmented naive Bayes for ranking. In L. Zhou, B.C. Ooi & X. Meng, eds., *DASFAA*, vol. 3453 of *Lecture Notes in Computer Science*, 688–698, Springer. 8

John, G.H. & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 338–345. 30, 102

Karciauskas, G. (2005). Learning with hidden variables: A parameter reusing approach for tree-structured Bayesian networks. *Ph.D. thesis*. 64, 159

Keogh, E. & Pazzani, M. (1999). Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *Proceedings of the 7th International Workshop on AI and Statistics*, 225–230. 12

King, R.D., Feng, C. & Sutherland, A. (1995). Statlog: Comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, **9**, 289–333. 144

Kohavi, R. & Wolpert, D.H. (1996). Bias plus variance decomposition for zero-one loss functions. In *Machine Learning: Proceedings of the Thirteenth International*, 275–283, Morgan Kaufmann Publishers. 112

Kononenko, I. (1991). Semi-naive Bayesian classifier. In *Proceedings of the European working session on learning on Machine learning*, 206–219, Springer-Verlag New York, Inc., New York, NY, USA. 5

Korb, K. & Nicholson, A. (2010). *Bayesian Artificial Intelligence*. Chapman and Hall, 2nd edn. 9

LANGSETH, H. & NIELSEN, T.D. (2006). Classification using hierarchical naïve Bayes models. *Machine Learning*, **63**, 135–159. 48

LANGSETH, H., NIELSEN, T.D., RUMÍ, R. & SALMERÓN, A. (2009). Maximum likelihood learning of conditional MTE distributions. In *Proceedings of the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, ECSQARU '09, 240–251, Springer-Verlag, Berlin, Heidelberg. 34

LANGSETH, H., NIELSEN, T.D., RUMÍ, R. & SALMERÓN, A. (2010). Parameter estimation and model selection for mixtures of truncated exponentials. *Int. J. Approx. Reasoning*, **51**, 485–498. 34

LANGSETH, H., NIELSEN, T.D., RUMÍ, R. & SALMERÓN, A. (2012). Mixtures of truncated basis functions. *Int. J. Approx. Reasoning*, **53**, 212–227. 34

LARRAÑAGA, P., ETXEBERRIA, R., LOZANO, J. & PEÑA, J.M. (1999). Optimization by learning and simulation of Bayesian and Gaussian networks. Tech. rep., University of the Basque Country. 26

LAURITZEN, S.L. (1992). Propagation of probabilities, means, and variances in Mixed Graphical Association Models. *Journal of the American Statistical Association*, **87**, 1098–1108. 28

LAURITZEN, S.L. & JENSEN, F. (2001). Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, **11**, 191–203. 25, 28, 68

LEBOURGEOIS, F. & EMPTOZ, H. (1996). Pretopological approach for supervised learning. *International Conference on Pattern Recognition*, **4**, 256. 43

LEE, C.H. (2007). A Hellinger-based discretization method for numeric attributes in classification learning. *Knowledge-Based Systems*, **20**, 419–425. 20

LIM, T.S., LOH, W.Y. & SHIH, Y.S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, **40**, 203–228. 36

LIU, H., HUSSAIN, F., TAN, C.L. & DASH, M. (2002). Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery*, **6**, 393–423. 20

LOWD, D. & DOMINGOS, P. (2005). Naive Bayes models for probability estimation. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, 529–536, ACM, New York, NY, USA. 50, 54

LUCAS, P. (2004). Restricted Bayesian network structure learning. *Studies In Fuzziness And Soft Computing*, **49**, 217–232. 15

LUENGO, J. & HERRERA, F. (2009). Domains of competence of artificial neural networks using measures of separability of classes. In *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part I: Bio-Inspired Systems: Computational and Ambient Intelligence*, IWANN '09, 81–88, Springer-Verlag, Berlin, Heidelberg. 37, 121

LUENGO, J. & HERRERA, F. (2010a). Domains of competence of fuzzy rule based classification systems with data complexity measures: A case of study using a fuzzy hybrid genetic based machine learning method. *Fuzzy Sets and Systems*, **161**, 3–19. 37, 122

LUENGO, J. & HERRERA, F. (2010b). An extraction method for the characterization of the fuzzy rule based classification systems' behavior using data complexity measures: A case of study with FH-GBML. In *FUZZ-IEEE*, 1–8, IEEE. 124, 125, 130

LUENGO, J. & HERRERA, F. (2010c). Obtención de los dominios de competencia de C4.5 por medio de medidas de separabilidad de clases. In *Congreso Español de Informática 2010(TTIA 2010)*. 37, 122

MACIÀ, N., HO, T.K., ORRIOLS-PUIG, A. & BERNADÓ-MANSILLA, E. (2010). The landscape contest at ICPR 2010. In D. Ünay, Z. Çataltepe &

S. Aksoy, eds., *ICPR Contests*, vol. 6388 of *Lecture Notes in Computer Science*, 29–45, Springer. 37

Malina, W. (2001). Two-parameter fisher criterion. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, **31**, 629–636. 39

Martínez, A.M., Webb, G.I., Flores, M.J. & Gámez, J.A. (2012). Non-disjoint discretization for aggregating one-dependence estimator classifiers. In E. Corchado, V. Snásel, A. Abraham, M. Wozniak, M. Graña & S.B. Cho, eds., *HAIS (2)*, vol. 7209 of *Lecture Notes in Computer Science*, 151–162, Springer. 107, 158

Michalski, R. & Chilausky, R. (1980). Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, **4:2**. 59

Michie, D., Spiegelhalter, D.J. & Taylor, C.C., eds. (1994). *Machine learning, neural and statistical classification*. Ellis Horwood, Upper Saddle River, NJ, USA. 144

Miranda, E.J.L. (2011). Selección de ejemplos para clasificación: un enfoque basado en la caracterización de los datos de entrada. *Ph.D. thesis*, universidad de Granada. 37

Mollineda, R.A., Sánchez, J.S. & Sotoca, J.M. (2005). Data characterization for effective prototype selection. In *Proceedings of the 2nd Iberian Conference on Pattern Recognition and Image Analysis*, 27–34, Springer. 37, 122

Moral, S., Rumí, R. & Salmerón, A. (2001). Mixtures of Truncated Exponentials in hybrid Bayesian networks. In *ECSQARU '01: Proceedings of the 6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, 156–167, Springer-Verlag, London, UK. 31, 76, 78

Moral, S., Rumí, R. & Salmerón, A. (2003). Approximating Conditional MTE Distributions by means of Mixed Trees. In *ECSQARU*, 173–183, ECSQARU 2003. 33

Morales, D.A., Bengoetxea, E. & Larrañaga, P. (2008). Selection of human embryos for transfer by Bayesian classifiers. *Computers in Biology and Medicine*, **38**, 1177–1186. 9

Neapolitan, R.E. (2003). *Learning Bayesian Networks*. Prentice Hall. 26

Nemenyi, P. (1963). Distribution-free multiple comparisons. *Ph.D. thesis*, Princeton University. 92

Orriols-Puig, A., Macià, N. & Ho, T.K. (2010). Documentation for the data complexity library in C++. Tech. rep., La Salle - Universitat Ramon Llull. 37, 38, 41, 122, 126

Pavlenko, T. & Chernyak, O. (2010). Credit risk modeling using Bayesian networks. *International Journal of Intelligent Systems*, **25**, 326–344. 9

Pérez, A., Larrañaga, P. & Inza, I. (2006). Supervised classification with conditional Gaussian networks: Increasing the structure complexity from naive Bayes. *International Journal of Approximate Reasoning*, **43**, 1–25. 27, 66

Pérez, A., Larrañaga, P. & Inza, I.n. (2009). Bayesian classifiers based on kernel density estimation: Flexible classifiers. *International Journal of Approximate Reasoning*, **50**, 341–362. 30

Platt, J.C. (1999). *Fast training of support vector machines using sequential minimal optimization*, 185–208. MIT Press, Cambridge, MA, USA. 41

Porwal, A., Carranza, E.J.M. & Hale, M. (2006). Bayesian network classifiers for mineral potential mapping. *Computers & Geosciences*, **32**, 1–16. 9

Qazi, M., Fung, G., Krishnan, S., Rosales, R., Steck, H., Rao, R.B., Poldermans, D. & Chandrasekaran, D. (2007). Automated heart wall motion abnormality detection from ultrasound images using Bayesian networks. In *Proceedings of the 20th international joint conference on Artificial intelligence*, IJCAI'07, 519–525, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 9

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, **1**, 81–106. 33

Quinlan, J.R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 144

Read, J., Pfahringer, B., Holmes, G. & Frank, E. (2009). Classifier Chains for multi-label classification. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, 254–269, Springer-Verlag, Berlin, Heidelberg. 150

Rehg, J.M., Murphy, K.P. & Fieguth, P.W. (1999). Vision-based speaker detection using Bayesian networks. In *CVPR*, 2110–2116, IEEE Computer Society. 8

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, **14**, 465–471. 53

Rubio, A. & Gámez, J.A. (2011). Flexible learning of k-dependence Bayesian network classifiers. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, 1219–1226, ACM, New York, NY, USA. 99

Rumí, R., Salmerón, A. & Moral, S. (2006). Estimating Mixtures of Truncated Exponentials in hybrid Bayesian networks. *TEST: An Official Journal of the Spanish Society of Statistics and Operations Research*, **15**, 397–421. 32, 78, 79, 80

RUSSELL, S. & NORVIG, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edn. 8

SAHAMI, M. (1996). Learning limited dependence Bayesian classifiers. In *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases*, 335–338. 15, 90, 99, 100

SAHAMI, M., DUMAIS, S., HECKERMAN, D. & HORVITZ, E. (1998). A Bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, AAAI Technical Report WS-98-05, Madison, Wisconsin. 8

SÁNCHEZ, J.S., MOLLINEDA, R.A. & SOTOCA, J.M. (2007). An analysis of how training data complexity affects the nearest neighbor classifiers. *Pattern Analysis and Applications*, **10**, 189–201. 37, 122

SCHWARZ, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, **6**, 461–464. 34

SHENOY, P. & WEST, J. (2009). Mixtures of polynomials in hybrid Bayesian networks with deterministic variables. In *Proceedings of the 8th Workshop on Uncertainty Processing*, WUPES-09, 202–212. 34

SHENOY, P.P. (2011). A re-definition of mixtures of polynomials for inference in hybrid Bayesian networks. In *Proceedings of the 11th European conference on Symbolic and quantitative approaches to reasoning with uncertainty*, ECSQARU'11, 98–109, Springer-Verlag, Berlin, Heidelberg. 34

SOHN, S.Y. (1999). Meta analysis of classification algorithms for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**, 1137–1144. 36, 145

SU, J. & ZHANG, H. (2006). Full Bayesian network classifiers. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, 897–904, ACM, New York, NY, USA. 17

TOH, K.A. (2008). An error-counting network for pattern classification. *Neurocomputing*, **71**, 1680–1693. 36

TSOUMAKAS, G. & KATAKIS, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, **3**, 1–13. 145

TSOUMAKAS, G. & VLAHAVAS, I. (2007). Random k-Labelsets: An ensemble method for multilabel classification. In J. Kok, J. Koronacki, R. Mantaras, S. Matwin, D. Mladenic & A. Skowron, eds., *Machine Learning: ECML 2007*, vol. 4701 of *Lecture Notes in Computer Science*, chap. 38, 406–417, Springer Berlin / Heidelberg, Berlin, Heidelberg. 147

TSOUMAKAS, G., KATAKIS, I. & VLAHAVAS, I. (2010). Mining multi-label data. In O. Maimon & L. Rokach, eds., *Data Mining and Knowledge Discovery Handbook*, chap. 34, 667–685, Springer US, Boston, MA. 148

TSOUMAKAS, G., SPYROMITROS-XIOUFIS, E., VILCEK, J. & VLAHAVAS, I. (2011). Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, **12**, 2411–2414. 149

VAN DER WALT, C. & BARNARD, E. (2006). Data characteristics that determine classifier performance. 166–171. 145

VAPNIK, V.N. (1995). *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA. 41

WANG, Q., GARRITY, G.M., TIEDJE, J.M. & COLE, J.R. (2007). Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Applied and Environmental Microbiology*, **73**, 5261–5267. 9

WEBB, G.I. (2000). Multiboosting: A Technique for Combining Boosting and Wagging. *Machine Learning*, **40**, 159–196. 92, 97

WEBB, G.I. & CONILIONE, P. (2002). Estimating bias and variance from data. 92, 112

WEBB, G.I., BOUGHTON, J.R. & WANG, Z. (2005). Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, **58**, 5–24. 5, 12, 90, 157

Webb, G.I., Boughton, J., Zheng, F., Ting, K.M. & Salem, H. (2012). Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification. *Machine Learning*, **86**, 233–272. 118

Weiss, N.A. (2002). *Introductory statistics*. Greg Tobin, 6th edn. 102

WEKA-Datasets (2008). Collection of datasets avalaibles from WEKA's official homepage. http://www.cs.waikato.ac.nz/ml/weka/. 54, 70

Witten, I.H. & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edn. 13, 21

Wolpert, D.H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, **8**, 1341–1390. 36

Wong, T.T. (2012). A hybrid discretization method for naïve Bayesian classifiers. *Pattern Recognition*, **45**, 2321–2325. 20

Yang, Y. (2003). *Discretization for Naive-Bayes Learning Ph.D. Thesis*. Monash University, http://www.csse.monash.edu.au/~webb/Files/Yingthesis.pdf. 19

Yang, Y. & Webb, G.I. (2001). Proportional k-interval discretization for naive-Bayes classifiers. In *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*, 564–575, Springer-Verlag, London, UK. 24, 101, 102, 111

Yang, Y. & Webb, G.I. (2002). Non-disjoint discretization for naive-Bayes classifiers. In C. Sammut & A. Hoffmann, eds., *Proceedings of the Nineteenth International Conference on Machine Learning (ICML '02)*, 666–673, Morgan Kaufmann, San Francisco. 21, 24, 106, 107, 111, 112, 117

Yang, Y. & Webb, G.I. (2009). Discretization for Naive-Bayes Learning: Managing Discretization Bias and Variance. *Machine Learning*, **74**, 39–74. 20, 84, 97, 101, 103, 107

Zhang, H. & Lu, Y. (2002). Learning Bayesian network classifiers from data with missing values. In *TENCON '02. Proceedings of the IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol. 1, 35 – 38 vol.1. 12

Zhang, H., Jiang, L. & Su, J. (2005). Hidden naive Bayes. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 2*, 919–924, AAAI Press. 17, 48

Zheng, F. & Webb, G.I. (2005). A Comparative Study of Semi-naive Bayes Methods in Classification Learning. In S. Simoff, G. Williams, J. Galloway & I. Kolyshkina, eds., *Proceedings of the 4th Australasian Data Mining conference (AusDM05)*, 141–156. 5, 13

Zheng, Z. & Webb, G.I. (2000). Lazy Learning of Bayesian Rules. *Machine Learning*, **41**, 53–84. 12