Departamento de Ciencias de la Computación e Inteligencia Artificial
Konputazio Zientziak eta Adimen Artifiziala Saila
Department of Computer Science and Artificial Intelligence

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea
University of the Basque Country

Advances on Supervised and Unsupervised Learning of Bayesian Network Models. Application to Population Genetics

by

Guzmán Santafé

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

Donostia - San Sebastián, September 2007

Acknowledgements

On finishing this dissertation I can look back and see the long way that has led me here, and I recognise all the people that have helped and encouraged me throughout these years. Now, I would like to make use of these lines to acknowledge their never faltering support.

First, I would like to thank Jose A. Lozano and Pedro Larrañaga, my thesis advisors. I am deeply indebted to them for their constant help, encouragement and friendly guidance. They are the example to follow in my career.

I am also grateful to Iñaki Inza and the rest of my collages in the Intelligent Systems Group: Alex Mendiburu, Endika Bengoetxea, Teresa Mikelez, Rosa Blanco, Ramón Sagarna, Roberto Santana, Aritz Perez, Ruben Armañanzas, Borja Calvo, Juan Diego Rodríguez, José Luis Flores, Dinora Morales and Yosu Galdiano. All of them have contributed to the excellent working atmosphere in the research group and offered me their help and advice when it was necessary.

This dissertation would not be possible with the financial support offered by the Ministerio de Ciencia y Tecnología (under grant TIC2001-2973-C05-03), the University of the Basque Country (under grant 9/UPV 00140.226-15334/2003) and the Gobierno de Navarra (under grant *Ayudas para la elaboración de la tesis doctoral y la obtención del título de doctor*).

My most sincere gratitude to Xiaohui Liu, who hosted me in the Intelligent Data Analysis (IDA) group at the University of Brunel, West London, and to the rest of people from the IDA group. The experience there is somehow reflected in this work.

I am and will always be in debt to my parents and my family, who, by example, have taught me how to live and act. They have always supported and encouraged me and they are responsible for the person I am.

Finally, all my love to Edurne. Thanks for everything throughout the time you have been with me, standing by me through the good and the bad times, and always supporting me to the end of this adventure.

Contents

List of Figures

List of Tables

Part I

Introduction

1

Introduction

Supervised classification and data clustering are two fundamental disciplines of data mining and machine learning. Roughly speaking, supervised classification can be seen as learning from experience. The supervised classification task uses data where the class or the group structure is known in order to learn a model which is able to classify unseen data samples where the class is unknown. Conversely, data clustering directly aims to model the underlying (and unknown) group structure of the data.

Probabilistic graphical models (Pearl, 1988; Whittaker, 1991; Lauritzen, 1996) and particularly Bayesian networks (Castillo *et al.*, 1997; Jensen, 2001; Neapolitan, 2003; Jensen and Nielsen, 2007) are powerful probabilistic tools that have become very popular paradigms to represent uncertainty. These Bayesian network models can be learned to solve both supervised classification (Friedman *et al.*, 1997; Larrañaga *et al.*, 2005) and clustering problems (Cheeseman and Stutz, 1996).

This dissertation aims to contribute to the state of the art of both supervised classification and data clustering disciplines by providing new algorithms to learn Bayesian networks. In the following sections we clarify these contributions, which are introduced throughout the dissertation.

1.1 Contributions of the Dissertation

The contributions introduced in this dissertation are presented in two main parts. On the one hand, the contributions related to supervised classification are focused on the discriminative learning of Bayesian network classifiers. On the other hand, the part related to data clustering introduces new methods to deal with Bayesian model averaging for clustering.

1.1.1 Discriminative Learning of Bayesian Network Classifiers

Machine learning approaches for supervised classification problems have generally fallen into two major categories: generative and discriminative methods. Generative methods approach the classification problem by modeling the joint probability distribution over the data samples. By contrast, discriminative methods are focused on learning the conditional relation of the class label given the rest of the variables. Bearing Bayesian network classifiers in mind, the generative learning of the classification model is obtained by maximizing the likelihood of the dataset given the model. Conversely, in order to learn a Bayesian network classifier from a discriminative approach, the learning method should maximize the conditional likelihood of the class label given the rest of the variables.

The factorization of the joint probability distribution given by Bayesian network models allows the decomposition of the calculations for the likelihood function leading to a simple and efficient computation. Unfortunately, although a discriminative approach is apparently a more natural way to learn classification models because it is oriented to classification, in the case of Bayesian network classifiers, discriminative learning is much more inefficient than generative learning. Nevertheless, several methods for an efficient discriminative learning of Bayesian network classifiers have been proposed in the literature (Huang *et al.*, 2003; Grossman and Domingos, 2004; Huang *et al.*, 2005; Roos *et al.*, 2005; Greiner *et al.*, 2005; Guo and Greiner, 2005; Pernkopf and Bilmes, 2005; Feelders and Ivanovs, 2006; Perez *et al.*, 2006).

Chapter 5 motivates the use of a discriminative approach to learn Bayesian network classifiers. However, the main contribution of this chapter is the adaptation of the TM algorithm (Edwards and Lauritzen, 2001) to learn the parameters of Bayesian network classifiers by maximizing the conditional likelihood and also the extension of the TM algorithm to learn the structure of Bayesian network classifiers. Additionally, we also present experimental results and empirical evidence to show when the discriminative approach should be preferred to the generative one.

1.1.2 Bayesian Model Averaging of Bayesian Network Models for Clustering

Typical approaches to learn Bayesian networks usually attempt to maximize some quantity, such as the likelihood, in order to obtain the best model describing the data. However, these approaches neglect the uncertainty in model selection. By contrast, the Bayesian approach treats every uncertain quantity as a random variable and uses the laws of probability to manipulate these uncertain quantities. The Bayesian approach averages over the possible settings of all uncertain quantities rather than selecting a single configuration for these quantities. Although the Bayesian approach is a proper manner to deal with

uncertainty, usually Bayesian model averaging calculations are computationally intractable and only approximations are feasible.

The majority of the methods for the Bayesian model averaging of Bayesian networks are proposed for supervised classification tasks, where the dataset does not contain missing values (Dash and Cooper, 2002; Cerquides and López de Mántaras, 2003a; Dash and Cooper, 2003; Cerquides and López de Mántaras, 2003b, 2005; Dash and Cooper, 2004; Hwang and Zhang, 2005). Conversely, in the data clustering problem, the presence of hidden variables prevents the exact computation of Bayesian model averaging calculations. The approximations to Bayesian model averaging of Bayesian network models in the presence of hidden variables are usually based on the calculation of the marginal likelihood by means of stochastic simulation or Laplace's approximation (Chickering *et al.*, 1995).

Chapter 7 overviews the Bayesian model averaging problem and introduces the main contribution of the dissertation in the field of data clustering. This contribution is given by the Expectation Model Averaging algorithm for naive Bayes (EMA) and for TAN models (EMA-TAN). The EMA and EMA-TAN algorithms presented in this dissertation allow to approximate, under certain restrictions, the Bayesian model averaging of naive Bayes and TAN models, respectively, for clustering in an efficient manner. In the same chapter the characteristics of the proposed algorithms are empirically evaluated using different datasets.

In addition to the theoretical development of the EMA and EMA-TAN algorithms and their empirical evaluation, Chapter 8 presents an application of the EMA algorithm to a real problem taken from population genetics. In this chapter, the EMA algorithm is modified to take into account the special characteristics of the problem and additionally, a method for unsupervised feature selection based on the model learned by the EMA algorithm is proposed.

1.2 Overview of the Dissertation

This dissertation is divided into nine chapters, which are organized into five parts. The first part consist of two chapters. This first chapter is an introduction to the dissertation where the reader can find a synthesis of the contributions and how the dissertation is structured. Chapter 2 introduces probabilistic graphical models with special attention on Bayesian network models. In this chapter, the basic notation used throughout the dissertation is presented and some relevant concepts are reviewed.

Part II is devoted to the supervised classification problem. Chapter 3 is a general introduction to the problem, paying special attention to the evaluation of the classifiers and the comparison between different classifiers. In Chapter 4, some Bayesian network classifiers (naive Bayes, selective naive Bayes, tree augmented naive Bayes, k-dependence Bayesian classifiers and Bayesian network augmented naive Bayes) are exposed. These Bayesian network classifiers

are used in the following chapters of the dissertation and in the derivation of new learning algorithms. Chapter 5 presents the differences between generative and discriminative learning. In this chapter, novel methods to learn the parameters and the structure of Bayesian network classifiers from a discriminative point of view are introduced.

Part III is focused on data clustering. Chapter 6 presents the data clustering problem and briefly reviews several clustering methods with special attention on methods to learn Bayesian networks for clustering. Chapter 7 reviews the Bayesian model averaging to learn Bayesian network models and introduces novel algorithms to approximate the Bayesian model averaging for clustering.

Part IV, and consequently Chapter 8, presents an application of the EMA algorithm, one of the Bayesian model averaging methods for clustering introduced in Chapter 7. Finally, Part V (Chapter 9) summarizes the work in the dissertation, the publications and the future work.

2

Probabilistic Graphical Models

Probabilistic graphical models (PGMs) (Pearl, 1988; Whittaker, 1991; Lauritzen, 1996; Castillo *et al.*, 1997) are powerful probabilistic tools that have become a very popular paradigm to represent uncertain knowledge in expert systems. PGMs are composed of two main parts: structure and parameters. The structure of a PGM can be represented by a variety of frameworks including undirected graphs, directed graphs and chain graphs among others and it is related to the conditional (in)dependence relationships among the variables. The presence of a link between two nodes represents the existence of a conditional dependency relationship between the variables represented by those nodes. By contrast, the absence of links can be related to conditional independence relationships between variables. The parameters of the PGMs are usually represented by conditional and/or marginal probabilities and they, in some way, characterize the strength of the dependencies defined in the graph structure.

In this chapter, we introduce the general notation used throughout the dissertation and some basic concepts about PGMs based on directed acyclic graphs. Then, we focus on Bayesian networks, a well-known probabilistic graphical model which is crucial for the development of this work.

2.1 Notation

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

The generalized joint probability distribution of $\boldsymbol{X}$ over a point $\boldsymbol{x}$ is given by $\rho(\boldsymbol{X} = \boldsymbol{x})$ or just by $\rho(\boldsymbol{x})$. $\rho(x)$ denotes the marginal generalized probability distribution of X and we use $\rho(x_i|x_j)$ to represent the conditional generalized distribution of X_i given $X_j = x_j$.

When a random variable can have a numerable number of values, we will refer to it as a discrete random variable otherwise, if it can have a non-numerable number of values, we will refer to it as a continuous random variable. If every unidimensional random variable in a n-dimensional set of random variables is a discrete random variable, $\rho(\boldsymbol{x}) = p(\boldsymbol{x})$ is known as the joint probability mass function of $\boldsymbol{X}$. Similarly, $p(x)$ and $p(x_i|x_j)$ represent the marginal and the conditional probability mass functions respectively. The term probability distribution is sometimes used in the literature to refer to a probability mass function. Although, strictly speaking, the latter is more correct than the former, both terms are used indistinctly throughout this dissertation. By contrast, if the random variables in $\boldsymbol{X}$ are continuous, $\rho(\boldsymbol{x}) = f(\boldsymbol{x})$ is the joint density function of $\boldsymbol{X}$, and $f(x)$ and $f(x_i|x_j)$ denote the marginal and conditional density functions respectively.

2.2 Probabilistic Graphical Models Based on Directed Acyclic Graphs

A probabilistic graphical model based on a directed acyclic graph is a representation of the joint generalized probability distribution, $\rho(\boldsymbol{x})$. In this case, the structure of the PGM, S, is given by a directed acyclic graph which describes a set of conditional independence relationships among the random variables in $\boldsymbol{X}$. The graph S defines, for each random variable X_i, the set of parents $\boldsymbol{Pa}_i^S$ with $i = 1, \ldots, n$. Thus, X_i and $\{X_1, \ldots, X_n\} \setminus \boldsymbol{Pa}_i^S$ are conditionally independent given $\boldsymbol{Pa}_i^S$. Throughout the dissertation, when it is clear from the context that $\boldsymbol{Pa}_i$ is defined by a specific structure S we avoid the use of the super-script S for a simpler notation.

The probabilistic graphical model represents a factorization of the generalized probability which, due to the chain rule, can be written as:

$$\rho(\boldsymbol{x}) = \prod_{i=1}^{n} \rho(x_i|\boldsymbol{pa}_i) \tag{2.1}$$

Apart from the structure S, a probabilistic graphical model defines another component, the set of parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. Therefore, Equation 2.1 can be rewritten as follows:

$$\rho(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{i=1}^{n} \rho(x_i|\boldsymbol{pa}_i, \boldsymbol{\theta}) \tag{2.2}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$. Taking both components of the probabilistic graphical model into account, it can be represented as a pair $\mathcal{M} = (S, \boldsymbol{\theta})$.

In order to understand the semantics of a probabilistic graphical model, the conditional independence criterion is essential.

Definition 1. *Let $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$ be three disjoint sets of variables. $\boldsymbol{Y}$ is conditionally independent of $\boldsymbol{Z}$ given $\boldsymbol{W}$ if and only if*

$$p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{w}) = p(\boldsymbol{y}|\boldsymbol{w}) \tag{2.3}$$

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

When $\boldsymbol{Y}$ is conditionally independent of $\boldsymbol{Z}$ given $\boldsymbol{W}$ it is denoted by $CI(\boldsymbol{Y}, \boldsymbol{Z}|\boldsymbol{W})$. Informally, the fact that $\boldsymbol{Y}$ is conditionally independent of $\boldsymbol{Z}$ given $\boldsymbol{W}$ means that the knowledge about the value of $\boldsymbol{Z}$ provides no information about $\boldsymbol{Y}$ if the value of $\boldsymbol{W}$ is already known. In probabilistic graphical models based on directed acyclic graphs, the conditional independence assertion is related to the d-separation criterion.

Definition 2. *Let $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$ be three disjoint sets of variables. $\boldsymbol{Y}$ is conditionally independent of $\boldsymbol{Z}$ given $\boldsymbol{W}$ in a probabilistic graphical structure, S, if $\boldsymbol{W}$ d-separates $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in S.*

Although several definitions of d-separation exist, in this dissertation we present the one given by Lauritzen *et al.* (1990) because, in our opinion, it provides a more intuitive interpretation of the concept. This definition is based on u-separation criterion and undirected graphs. For other definitions of d-separation see Pearl (1988) and Lauritzen *et al.* (1990).

Definition 3. *Let G be an undirected graph and let $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$ be three disjoint sets of variables. $\boldsymbol{W}$ u-separates $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in G if every path in G between a node belonging to $\boldsymbol{Y}$ and a node belonging to $\boldsymbol{Z}$ contains at least one node belonging to $\boldsymbol{W}$.*

In order to introduce the definition of the d-separation criterion, the definitions of ancestral set of nodes and moral graph are needed:

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

Definition 5. *Let G be a graph and Y and Z be two variables. Z is an ancestral node of Y in G if there is a directed path between Z and Y in G.*

Definition 6. *Let G be a directed acyclic graph, the moral graph associated to G is the graph obtained by adding an arc between parents with a common child and then making all arcs in G undirected, that is turning the arcs into edges.*

Then, we are able to define the d-separation criterion on the basis of the u-separation criterion.

Definition 7. *Let S be a structure of a probabilistic graphical model represented by a directed acyclic graph and let $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$ be three disjoint sets of variables. $\boldsymbol{W}$ d-separates $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in S if $\boldsymbol{W}$ u-separates $\boldsymbol{Y}$ and $\boldsymbol{Z}$ in the moral graph of the smallest ancestral set of nodes which contains $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$.*

2.3 Bayesian Networks

Bayesian networks are probabilistic graphical models based on directed acyclic graphs where the nodes are discrete random variables. Over the last years, Bayesian networks have received considerable attention from the machine learning community. As a result of this interest, many theoretical publications have appeared. Pearl (1988) is a well-known reference, but there are other relevant tutorials such as Castillo *et al.* (1997), Jordan (1998), Neapolitan (2003), Korb and Nicholson (2004) and Jensen and Nielsen (2007), or Lauritzen (1996) which provides a mathematical analysis of graphical models. Additionally, Bayesian networks have been used for probabilistic inference in different domains such as expert system (Dawid, 1992; Lauritzen and Spiegelhalter, 1988), classification problems (Friedman *et al.*, 1997; Larrañaga *et al.*, 2005), optimization (Mühlenbein and Mahning, 2001; Larrañaga and Lozano, 2002) or bioinformatics (Friedman and Pe'er, 2000; Larrañaga *et al.*, 2006).

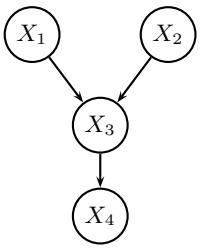
2.3.1 Notation

In a Bayesian network, each variable $X_i \in \boldsymbol{X}$ represents a unidimensional discrete random variable with r_i possible states, $\{x_i^1, \ldots, x_i^{r_i}\}$, and the local distribution $p(x_i|\boldsymbol{pa}_i^j, \boldsymbol{\theta}_i)$ is a unrestricted discrete distribution:

$$p(x_i^k|\boldsymbol{pa}_i^j, \boldsymbol{\theta}_i) = \theta_{x_i^k|\boldsymbol{pa}_i^j} = \theta_{ijk} \tag{2.4}$$

where $\boldsymbol{pa}_i^1, \ldots, \boldsymbol{pa}_i^j, \ldots, \boldsymbol{pa}_i^{q_i}$ denote the q_i possible configurations that the set of parents $\boldsymbol{Pa}_i$ can take, with $q_i = \prod_{X_g \in \boldsymbol{Pa}_i} r_g$. A Bayesian network is denoted by a pair $\mathcal{B} = (S, \boldsymbol{\theta})$, where S is the structure of the model encoded by a directed acyclic graph and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$ is the set of parameters. The parameters of the local probability distribution for every X_i is given by $\boldsymbol{\theta}_i = ((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i}$ and θ_{ijk} represents the conditional probability of X_i taking its k-th state given that $\boldsymbol{Pa}_i$ takes its j-th configuration. Therefore, $\theta_{ijk} > 0$ and $\sum_{k=1}^{r_i} \theta_{ijk} = 1$.

Figure 2.1 shows an example of a Bayesian network model with four variables. The figure describes the structure and the parameters of the model as



Local probabilities:

$$
\begin{array}{ll}
\boldsymbol{\theta}_1 = (\theta_{1-1}, \theta_{1-2}) & p(x_1^1), p(x_1^2) \\
\boldsymbol{\theta}_2 = (\theta_{2-1}, \theta_{2-2}, \theta_{2-3}) & p(x_2^1), p(x_2^2), p(x_2^3) \\
\boldsymbol{\theta}_3 = (\theta_{311}, \theta_{321}, \theta_{331}, & p(x_3^1|x_1^1, x_2^1), p(x_3^1|x_1^1, x_2^2), p(x_3^1|x_1^1, x_2^3), \\
\quad \theta_{341}, \theta_{351}, \theta_{361}, & p(x_3^1|x_1^2, x_2^1), p(x_3^1|x_1^2, x_2^2), p(x_3^1|x_1^2, x_2^3), \\
\quad \theta_{312}, \theta_{322}, \theta_{332}, & p(x_3^2|x_1^1, x_2^1), p(x_3^2|x_1^1, x_2^2), p(x_3^2|x_1^1, x_2^3), \\
\quad \theta_{342}, \theta_{352}, \theta_{362}) & p(x_3^2|x_1^2, x_2^1), p(x_3^2|x_1^2, x_2^2), p(x_3^2|x_1^2, x_2^3), \\
\boldsymbol{\theta}_4 = (\theta_{411}, \theta_{421}, \theta_{412}, \theta_{422}) & p(x_4^1|x_3^1), p(x_4^1|x_3^2), p(x_4^2|x_3^1), p(x_4^2|x_3^2)
\end{array}
$$

Factorization of the joint mass probability function:

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2)p(x_3|x_1, x_2)p(x_4|x_3)$$

Fig. 2.1. Structure, local probabilities and factorization of the joint mass probability function for a Bayesian network with 4 variables, where X_1,X_3 and X_4 can take two values and X_2 three.

X_i	r_i	$\boldsymbol{Pa}_i$	q_i
X_1	2	$\emptyset$	0
X_2	3	$\emptyset$	0
X_3	2	$\{X_1, X_2\}$	6
X_4	2	$\{X_3\}$	2

Table 2.1. Description of the variables and the parent set for each variable of the Bayesian network model from Figure 2.1.

well as the factorization of the mass probability function derived from the given Bayesian network. Note that, while the joint probability distribution needs twenty-three parameters to encode the local conditional probabilities, the Bayesian network model shown in Figure 2.1 needs only eleven parameters. This reduction in the number of parameters is due to the factorization derived from the graphical structure of the model. Additionally, Table 2.1 shows the parent set for each variable and the number of possible configurations for each parent set. The current example, introduces a new notation, θ_{i-k}, which is used to denote the probability of $X_i = x_i^k$ given that X_i has no parents. This notation is also used throughout the dissertation.

2.3.2 Model Learning

In order to induce or learn a Bayesian network model it is necessary to set:

- A structure for the model by means of a directed acyclic graph which encodes the conditional (in)dependencies among the variables.
- A set of parameters which represents:
  - The unconditional local probabilities for the variables with no parents, θ_{i-k}.
  - The conditional local probabilities for all the variables with parents given all the possible configurations of the parents, θ_{ijk}.

The structure and the parameters needed to characterize a Bayesian network model can be provided externally by experts or they can be learned from data. Building a Bayesian network from expert knowledge is a very high consuming task and it is subject to mistakes. By contrast, learning a Bayesian network from data would be a more accurate and objective process since it attempts to obtain the (in)dependence relationships and conditional probabilities underlying the data. In the literature there are several good reviews of methods to learn Bayesian networks, for instance Heckerman (1995), Heckerman *et al.* (1995) and Neapolitan (2003) provide in-depth tutorials on learning Bayesian networks, Buntine (1996) presents a literature survey and Jordan (1998) collects several introductory surveys as well as papers discussing advances on learning Bayesian networks.

The learning process of a Bayesian network is usually divided into structure learning and parameter learning. One of the aspects that influences the complexity of a Bayesian network is the connectivity of the model structure. According to this complexity, a number of particular Bayesian network structures can be defined.

Definition 8. *A tree is a Bayesian network where each variable has only one parent, except for the root variable which has no parents.*

Definition 9. *A forest is a Bayesian network where each variable can have not more than one parent. This is a generalization of tree structures since it allows the Bayesian network to be formed by several unconnected trees.*

Definition 10. *A polytree is a Bayesian network that does not allow cycles in the associated undirected graph. That is, there is only one undirected path connecting any two nodes in the graph.*

Definition 11. *A multiple connected Bayesian network is a network structure where any two variables in the directed acyclic graph can be connected by more than one undirected path.*

In this section, we present a short overview of learning methods for Bayesian networks. First, we introduce some aspects of parameter learning. Although it is necessary to set a structure before learning the parameters of the Bayesian network, some aspects of parameter learning introduced in the following section are used for the learning of the structure. Therefore, we decide to present the parameter learning before the structure learning. Finally, we present the algorithms to learn the structure of a Bayesian network in two main groups: algorithms based on detecting conditional (in)dependencies and algorithms based on optimization or score+search methods. Additionally, some scores used in the learning of Bayesian networks are presented.

2.3.2.1 Parameter Learning

From a purely Bayesian approach, the correct manner to make predictions is by averaging over all the parameter configurations:

$$p(\boldsymbol{x}|D,S) = \int p(\boldsymbol{x}|D,S,\boldsymbol{\theta})p(\boldsymbol{\theta}|D,S)d\boldsymbol{\theta}$$
$$= \int p(\boldsymbol{x}|D,S,\boldsymbol{\theta})\frac{p(D|\boldsymbol{\theta},S)p(\boldsymbol{\theta}|S)}{p(D|S)}d\boldsymbol{\theta} \tag{2.5}$$

where D is the dataset. This approach is known as Bayesian model averaging. In the case of Bayesian network models, the Bayesian model averaging calculations can be efficiently obtained in closed form under three assumptions. The first assumption is that there is no missing data in the dataset D. The second assumption is that the parameter vectors $\boldsymbol{\theta}_{ij}$ are mutually independent. This assumption is known as parameter independence (Spiegelhalter and Lauritzen, 1990):

$$p(\boldsymbol{\theta}|S) = \prod_{i=1}^{n}\prod_{j=1}^{q_i} p(\boldsymbol{\theta}_{ij}|S) \tag{2.6}$$

Finally, the last assumption states Dirichlet *priors* over the parameters:

$$p(\boldsymbol{\theta}_{ij}|S) \sim Dir(\boldsymbol{\theta}_{ij}|\alpha_{ij1},\ldots,\alpha_{ijr_i}) = \frac{\Gamma(\alpha_{ij})}{\prod_{k=1}^{r_i}\Gamma(\alpha_{ijk})}\prod_{k=1}^{r_i}\theta_{ijk}^{\alpha_{ijk}-1} \tag{2.7}$$

where $\Gamma(.)$ is the gamma function, $(\alpha_{ij1},\ldots,\alpha_{ijr_i})$ are the hyperparameters of the Dirichlet distribution and $\alpha_{ij} = \sum_{k=1}^{r_i}\alpha_{ijk}$ for all i, j and k. Given these three assumptions, the Bayesian averaging over all the values of the parameters is equivalent to a single parameters set, $\breve{\boldsymbol{\theta}}$:

$$p(\boldsymbol{x}|D,S) = \prod_{i=1}^{n} p(x_i^k|\boldsymbol{pa}_i^j,D) = \prod_{i=1}^{n}\breve{\theta}_{ijk} \tag{2.8}$$

with

$$\breve{\theta}_{ijk} = \frac{N_{ijk}+\alpha_{ijk}}{N_{ij}+\alpha_{ij}} \tag{2.9}$$

where N_{ijk} is the number of data samples in D where X_i takes its k-th value and $\boldsymbol{Pa}_i$ takes its j-th configuration, and $N_{ij} = \sum_{k=1}^{r_i}N_{ijk}$ for all i,j and k.

By contrast to the Bayesian model averaging, other approaches select a single parameter configuration instead of averaging over all the possible configurations. The selected parameter configuration is then given as an estimator of the true probability distribution that generated the dataset. Two well-known approaches are the *maximum likelihood* (ML) estimation and the *maximum a posteriori* (MAP) estimation:

- *Maximum likelihood* estimation selects the parameter configuration for a Bayesian network model, $\hat{\boldsymbol{\theta}}$, that maximizes the probability of the dataset given the model:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(D|S, \boldsymbol{\theta}) \tag{2.10}$$

  with $p(D|S, \boldsymbol{\theta})$ being the likelihood function

$$p(D|S, \boldsymbol{\theta}) = \prod_{d=1}^{N} p(\boldsymbol{x}^{(d)}|S, \boldsymbol{\theta}) \tag{2.11}$$

  $\boldsymbol{x}^{(d)}$ being the value of $\boldsymbol{X}$ given by the d-th sample in the dataset D. Then, by maximizing the likelihood function we obtain the ML parameters as follows:

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}} \tag{2.12}$$

- *Maximum a posteriori* estimation selects the parameter configuration for a Bayesian network model, $\tilde{\boldsymbol{\theta}}$, that maximizes the posterior probability of the parameters:

$$\tilde{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|D, S) \tag{2.13}$$

  In the case of Bayesian networks, under the three assumptions introduced before, the MAP parameter calculation is equivalent to the Bayesian model averaging calculation given in Equation 2.9. However, both learning methods are fundamentally different approaches to learn the parameters from data.

A more detailed derivation of the parameters can be consulted in Heckerman (1995), Bernardo and Smith (2000) and MacKay (2003).

2.3.2.2 Structure Learning Algorithms Based on Detecting Conditional (In)Dependencies

A good review of structure learning algorithms based on detecting conditional (in)dependencies is provided by Spirtes *et al.* (1993) or de Campos (1998). This type of algorithm usually starts from a set of conditional independence relationships between subsets of variables and uses independence tests in order to learn a directed acyclic graph that represents a large percentage (and even all of them if possible) of the relationships captured in the data.

Two examples of learning algorithms based on detecting conditional (in)-dependencies are, for instance, the PC algorithm (Spirtes *et al.*, 1991), which is probably the best known algorithm to learn Bayesian network structures by detecting conditional independencies, and the algorithm to learn polytrees presented in Acid and de Campos (1995).

2.3.2.3 Structure Learning Algorithms Based on Score+Search Methods

The score+search approach deals with structure learning as an optimization problem. This structure learning of a Bayesian network requires, as does any other optimization problem, a score to measure the goodness of a candidate solution in the search space and a search strategy to explore this space. Any search strategy can be adapted to learn the structure of a Bayesian network. Therefore, some of the most relevant and most used heuristic searching methods have been applied to the structure learning of Bayesian networks. Greedy search (Buntine, 1991; Cooper and Herskovits, 1992), simulated annealing (Chickering *et al.*, 1995), tabu search (Bouckaert, 1995), genetic algorithms (Larrañaga *et al.*, 1996; Etxeberria *et al.*, 1998), estimation of distribution algorithms (Blanco *et al.*, 2003), Markov chain Monte Carlo (Myers *et al.*, 1999), variable neighborhood search (de Campos and Puerta, 2001) or ant colony optimization (de Campos *et al.*, 2002) are examples of searching methods used to learn the structure of Bayesian networks. By contrast, the scores used to guide the learning process must be tailored to the particularities of this optimization problem. In the following paragraphs we review some of the most used scores.

A. Marginal Likelihood

Marginal likelihood is a well-known score to measure the quality of a Bayesian network structure. The marginal likelihood is defined as the probability of a dataset D given a structure S. It is considered a Bayesian score because the calculation of the probability value averages over all the possible parameter configurations for the given structure S.

$$p(D|S) = \int p(D|S, \boldsymbol{\theta}) p(\boldsymbol{\theta}|S) d\boldsymbol{\theta} \tag{2.14}$$

Under the following assumptions:

1. All the variables in the dataset D are multinomial variables.
2. The samples in the dataset D are independent and identically distributed.
3. The dataset D is complete (there are no missing values).
4. Parameter independent assumption (see Equation 2.6).
5. The *prior* probability distribution over the parameters of the Bayesian network model given the structure S is a uniform distribution.

the marginal likelihood of the dataset D given the Bayesian network structure S can be calculated in closed form (Cooper and Herskovits, 1992):

$$p(D|S) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{2.15}$$

where N_{ijk} is the number of data samples in D so that X_i takes its k-th value and $\boldsymbol{Pa}_i$ takes its j-th configuration, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ for every i,j and k.

The closed form for the marginal likelihood defined in Equation 2.15 is known as $K2$ score. Moreover, the greedy algorithm that uses this score to learn the structure of a Bayesian network is also known as $K2$ algorithm (Cooper and Herskovits, 1992). The $K2$ algorithm assumes that an ancestral order among the variables is known and that every structure for the Bayesian network model is equally likely *a priori*. Additionally, the complexity of the resultant Bayesian network structure can be restricted by setting the maximum number of parents that every variable can have. The algorithm starts with a directed acyclic graph with no arcs. Then, for every variable X_i, the algorithm updates, at each time, the set of parents of X_i by adding the node that increases the score the most from those that appear before X_i in the provided ancestral ordering. The process finishes when no addition of a single parents improves the score. Obviously, this approach does not guarantee that the final structure is globally optimal with respect to the score.

The $K2$ metric could be criticized for assuming a *prior* uniform distribution over the Bayesian network parameters. The parameters of a Bayesian network represent multinomial probability distributions, and the conjugate *prior* of the multinomial distribution is the Dirichlet distribution (Raiffa and Schalaifer, 1961; Bernardo and Smith, 2000). This leads Heckerman *et al.* (1995) to derive the Bayesian Dirichlet metric. This metric is a generalization of the $K2$ metric that considers a *prior* Dirichlet distribution over the parameters of the Bayesian network model. For the derivation of the Bayesian Dirichlet metric we make the same assumptions as in the derivation of the $K2$ metric except for the uniform *prior* probability distribution over the parameters. Instead of a uniform distribution, we assume that the *prior* probability distribution over the parameters of the Bayesian network model given the structure S is a Dirichlet distribution with hyperparameters $(\alpha_{ij1}, \ldots, \alpha_{ijr_i})$

$$p(\boldsymbol{\theta}_{ij}|\alpha_{ij1}, \ldots, \alpha_{ijr_i}) \sim Dir(\alpha_{ij1}, \ldots, \alpha_{ijr_i}) \tag{2.16}$$

Therefore, the Bayesian Dirichlet score can be calculated in closed form as:

$$p(D|S) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})} \tag{2.17}$$

where $\Gamma(.)$ is the gamma function and $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$ for all i,j and k. The simplest approach to the Bayesian Dirichlet score is to set non-informative *priors*, that is, every parameter configuration is equally likely *a priori* ($\alpha_{ijk} = 1$). Note that this approach is equivalent to the $K2$ score defined by Cooper and Herskovits (1992). In addition, Heckerman *et al.* (1995) introduce a specialization of the Bayesian Dirichlet score named likelihood-equivalent Bayesian Dirichlet metric which is identical to the Bayesian Dirichlet score defined

in Equation 2.17 but making another new assumption, likelihood equivalence, which states that data should not help to discriminate between any two structures that represent the same conditionally independent statements.

B. Penalized log-Likelihood

Given a dataset D, one can calculate the *maximum likelihood* parameters $\hat{\boldsymbol{\theta}}$ for any Bayesian network structure S. Then, the likelihood score measures the probability of the dataset D given a Bayesian network model. For convenience in the calculations, the logarithm of the likelihood score is usually taken:

$$\begin{aligned}\log p(D|S,\boldsymbol{\theta}) &= \log \prod_{d=1}^{N} p(\boldsymbol{x}^{(d)}|S,\boldsymbol{\theta}) \\ &= \log \prod_{d=1}^{N}\prod_{i=1}^{n} p(x_i^{(d)}|\boldsymbol{pa}_i^{(d)},S,\boldsymbol{\theta}) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} \log(\theta_{ijk})^{N_{ijk}} \end{aligned} \tag{2.18}$$

where $\boldsymbol{x}^{(d)}$, $x_i^{(d)}$ and $\boldsymbol{pa}_i^{(d)}$ are the values of $\boldsymbol{X}$, X_i and $\boldsymbol{Pa}_i$ given by the d-th data sample in D respectively. Using the *maximum likelihood* estimation for the parameters, $\hat{\boldsymbol{\theta}}$, the formulation is as follows:

$$\log p(D|S,\hat{\boldsymbol{\theta}}) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\frac{N_{ijk}}{N_{ij}} \tag{2.19}$$

The use of the log-likelihood score in the searching process of the structure for a Bayesian network model has two main problems:

- The log-likelihood is a monotonous increasing function with respect to the complexity of the model structure. Therefore, the use of the log-likelihood as a score to evaluate the quality of a structure in the searching process usually leads the search towards complete Bayesian network structures.
- As the network complexity increases (the number of parents for each node increases) the error in the parameter estimation also increases.

In order to overcome these difficulties, a common solution is the addition of a penalty term to the log-likelihood calculation. A general formulation of the penalized log-likelihood is given by:

$$\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\frac{N_{ijk}}{N_{ij}} - \varphi(N)dim(S) \tag{2.20}$$

where $dim(S)$ is the dimension of the Bayesian network with structure S. Usually, the number of parameters needed to totally specify the Bayesian network is used to measure its dimension:

$$dim(S) = \sum_{i=1}^{n} q_i(r_i - 1) \tag{2.21}$$

On the other hand, $\varphi(N)$ is a non-negative penalty function. A very popular penalized log-likelihood score is the Bayesian Information Criterion (BIC) (Schwarz, 1978) where $\varphi(N) = \frac{1}{2}\log N$:

$$BIC = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\frac{N_{ijk}}{N_{ij}} - \frac{1}{2}\log N\sum_{i=1}^{n} q_i(r_i - 1) \tag{2.22}$$

Another example of penalized log-likelihood score is Akaike's Information Criterion (AIC) (Akaike, 1974), where $\varphi(N) = 1$.

C. Information Theory Scores

Scores that compare two joint probability distributions are called scoring rules. $Sc(p(\boldsymbol{x})||p'(\boldsymbol{x}))$ denotes the scoring rule that compares the true joint probability distribution $p(\boldsymbol{x})$ and the one approximated, in this case, by the Bayesian network $p'(\boldsymbol{x})$. A score is called a proper scoring rule if $Sc(p(\boldsymbol{x})||p(\boldsymbol{x})) \geq Sc(p(\boldsymbol{x})||p'(\boldsymbol{x}))$ for all $p'(\boldsymbol{x})$. Although there is an infinite number of functions that could be applied as a proper score (McCarthy, 1956), the logarithmic score has received special attention in the literature:

$$Sc(p(\boldsymbol{x})||p'(\boldsymbol{x})) = \sum_{\boldsymbol{x}} p(\boldsymbol{x})\log p'(\boldsymbol{x}) \tag{2.23}$$

The logarithmic scoring rule verifies the interesting property of being equivalent to the Kullback-Leibler divergence (Kullback and Leibler, 1951):

$$\begin{aligned} D_{KL}(p(\boldsymbol{x})||p'(\boldsymbol{x})) &= \sum_{\boldsymbol{x}} p(\boldsymbol{x})\log\frac{p(\boldsymbol{x})}{p'(\boldsymbol{x})} \\ &= \sum_{\boldsymbol{x}} p(\boldsymbol{x})\log p(\boldsymbol{x}) - \sum_{\boldsymbol{x}} p(\boldsymbol{x})\log p'(\boldsymbol{x}) \end{aligned} \tag{2.24}$$

The Kullback-Leibler divergence measures the difference between the true joint probability distribution, $p(\boldsymbol{x})$, and the approximation given by $p'(\boldsymbol{x})$. Nonetheless, although it is often intuited as a distance metric, the Kullback-Leibler divergence is not a true metric since it is not symmetric. Note that, as the term $\sum_{\boldsymbol{x}} p(\boldsymbol{x})\log p(\boldsymbol{x})$ from Equation 2.24 does not depend on the approximate probability distribution, $p'(\boldsymbol{x})$, the Kullback-Leibler divergence is a

linear transformation of the logarithmic scoring rule and then, maximizing the logarithmic scoring rule involves minimizing the Kullback-Leibler divergence.

Another approach related to the logarithmic scoring rule and to the Kullback-Leibler divergence is the minimum description length (MDL) principle (Rissanen, 1978). The MDL principle states that the best model structure to represent a dataset is the one that minimizes the sum of encoding lengths for the dataset and the model. The minimum description length can be seen as a penalized log-likelihood score or as a marginal likelihood score (Lam and Bacchus, 1994). However, it can be also obtained as a generalization of the Kullback-Leibler divergence. When applying the MDL principle to structure learning for Bayesian networks, the description length of the dataset can be calculated as $\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} -N_{ijk}\log\frac{N_{ijk}}{N_{ij}}$ and the length needed to store the parameters of the model is given by $\frac{1}{2}\log N dim(S)$. On the other hand, the encoding length for the structure is constant for all the Bayesian network structures with n nodes and therefore it is not needed. Thus, the MDL score is defined as follows:

$$MDL = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\frac{N_{ijk}}{N_{ij}} - \frac{1}{2}\log N dim(S) \tag{2.25}$$

Note that the MDL score is equivalent to the BIC score although their derivation processes are totally unrelated.

The literature includes several works where scores based on information theory such as the entropy (Herskovits and Cooper, 1990; Geiger, 1992) or mutual information (Chow and Liu, 1968) are used to learn the structure of Bayesian network models.

2.3.3 Inference and Simulation

Once a Bayesian network model is learned, it is considered a powerful mechanism for reasoning under uncertainty. The Bayesian network model is able to provide specific information about probability distributions of interest in the presence of information about the values of some other variables (evidence). In general, the computation of a probability of interest given a model is known as probabilistic inference and it can be calculated by marginalization over the rest of the variables, for instance, $p(x_n)$ can be computed as:

$$p(x_n) = \sum_{x_1}\dots\sum_{x_{n-1}} p(x_1,\dots,x_{n-1},x_n) \tag{2.26}$$

This approach is feasible for Bayesian network models with a small number of variables. However, in models with many variables the direct approach is not practical. Even when the conditional independencies encoded in the Bayesian

network model can be exploited to make the computation more efficient, exact probabilistic inference is NP-Hard (Cooper, 1990).

Many probabilistic algorithms for Bayesian network inference have been proposed. These proposals include exact probabilistic algorithms in tree or polytree structures (Pearl, 1986a; Lauritzen and Spiegelhalter, 1988; Jensen *et al.*, 1990b,a; Shachter, 1988; Díez, 1996) in arbitrary Bayesian networks (Lauritzen and Spiegelhalter, 1988; Jensen and Andersen, 1990; Dawid, 1992) or sampling methods such as Markov sampling methods (Pearl, 1986b), systematic sampling methods (Bouckaert, 1994) or the methods proposed by Chavez and Cooper (1990), Dagum and Luby (1993), Hryceij (1990), Kong and Kjærulff (1993) and Salmerón *et al.* (2000). Nonetheless, one of the most used methods for approximate probabilistic inference in Bayesian networks by simulation is the probabilistic logic sampling (PLS) (Henrion, 1988). This method takes an ancestral ordering among the variables according to the conditional independence relationships defined by the Bayesian network and then the values of the variables are sampled following the ancestral ordering. That is, a variable is sampled after all its parents have been sampled.

Part II

Supervised Classification

3

Introduction to Supervised Classification

Supervised classification is a part of machine learning with many applications in different fields such as bioinformatics, computer vision, speech recognition or medical diagnosis to name a few. Informally, supervised classification can be seen as learning from experience. It assumes the presence of a special variable called class variable which describes the value of the phenomenon of interest, for instance the diagnostic of the medical patient, the word representing a sound or the object represented in an image. The supervised classification task involves the use of data where the class variable is known in order to infer a model (classifier) which allows to label new data where the class value is unknown.

In this chapter we introduce the supervised classification problem as well as some methods used to evaluate and to compare classifiers obtained by means of different learning methods.

3.1 Supervised Classification Problem

Supervised classification can be structured in two main tasks. First, it is needed to learn the classification model (or classifier) from a dataset of N labeled instances $D = \{(c^{(1)}, \boldsymbol{x}^{(1)}), \ldots, (c^{(N)}, \boldsymbol{x}^{(N)})\}$, where each instance is characterized by n predictive variables, $\boldsymbol{X} = (X_1, \ldots, X_n)$, and the class, C, to which it belongs. Then, the learned classifier can be used to assess the class value $c \in \{1 \ldots, r_C\}$ of new instances characterized only by the predictive variables. Formally, a classifier can be seen as a function:

$$\gamma : (x_1, \ldots, x_n) \rightarrow \{1, \ldots, r_C\} \tag{3.1}$$

Each classification problem has an associated misclassification cost function, $cost(c_e, c_r)$ with $c_e, c_r = 1, \ldots, r_C$ and where c_r is the real class label and c_e is the class label estimated by the classifier. The misclassification cost

function provides the cost associated to classify a data sample of class c_r in a class c_e. The simplest, but widely used, misclassification cost function is the 0/1 loss function which is defined as:

$$cost(c_r, c_e) = \begin{cases} 0 & \text{if } c_r = c_e \\ 1 & \text{otherwise} \end{cases} \tag{3.2}$$

The goal of a good classifier is to minimize the total misclassification cost. Several classification paradigms have been developed in the past years including decision trees, decision rules, instance based classifiers, neural networks or support vector machines among many others. However, in this part of the dissertation we focus on Bayesian network classifiers, which are Bayesian network models used for supervised classification purposes. Bayesian network classifiers are a probabilistic approach to the supervised classification problem. They assume the existence of a joint probability distribution which generated the dataset:

$$p(c, x_1, \ldots, x_n) = p(c|x_1, \ldots, x_n)p(x_1, \ldots, x_n) \tag{3.3}$$

Given the joint probability distribution and the misclassification cost function $cost(c_r, c_e)$, the Bayesian network classifier that minimizes the total misclassification error can be obtained (Duda and Hart, 1973) by using the following assignment:

$$\gamma(\boldsymbol{x}) = \arg\min_k \sum_{c=1}^{r_C} cost(k, c)p(c|x_1, \ldots, x_n) \tag{3.4}$$

In the specific case of the 0/1 loss function, the classifier labels a given data instance with the maximum *a posteriori* class value, that is:

$$\gamma(\boldsymbol{x}) = \arg\max_c p(c|x_1, \ldots, x_n) = \arg\max_c p(c, x_1, \ldots, x_n) \tag{3.5}$$

In real problems, usually, the joint probability distribution is unknown, and it has to be estimated from the dataset.

The supervised classification paradigms can be structured in two different approaches: generative (or informative) and discriminative (Dawid, 1976; Rubinstein and Hastie, 1997). Generative classifiers model the joint probability distribution over all the variables including the class variable. That is, generative classifiers aim to model the probability distribution that generated the data. Then, the class conditional probability needed to classify new instances can be obtained by using the Bayes rule:

$$p(c|x_1,\ldots,x_n) = \frac{p(c,x_1,\ldots,x_n)}{\sum_{c'} p(c',x_1,\ldots,x_n)} \tag{3.6}$$

The parameters of the classification model are obtained by maximizing the log-likelihood (LL).

$$LL = \sum_{d=1}^{N} \log p(c^{(d)}, \boldsymbol{x}^{(d)}) \tag{3.7}$$

Examples of generative classifiers are, for instance, gaussian mixture models (McLachlan and Peel, 2000) and Bayesian network classifiers (Duda and Hart, 1973; Friedman *et al.*, 1997).

By contrast, discriminative classifiers directly model the posterior distribution of the class or just the class boundaries. Jebara (2003) introduces the term conditional learning in order to distinguish between discriminative classifiers based on probability distributions that model $p(c|x_1,\ldots,x_n)$, such as logistic regression (Hosmer and Lemeshow, 1989) or generalized additive models (Hastie and Tibshirani, 1990), and discriminative classifiers that only consider an input-output mapping, such as neural networks (Bishop, 1996) or support vector machines (Vapnik, 1998). In this dissertation, since we only consider probability-based models, with the term discriminative classifiers we will refer to the classifiers modeling the posterior distribution of the class. In this case, the parameters of the model are obtained by maximizing the logarithm of the conditional likelihood (CLL):

$$CLL = \sum_{d=1}^{N} \log p(c^{(d)}|\boldsymbol{x}^{(d)}) \tag{3.8}$$

3.2 Evaluation of Classifiers

A fundamental aspect of learning classification models is the evaluation of the resultant classifier (Mitchell, 1997). This evaluation or validation process aims to measure how the model behaves when classifying unseen data samples. A widely used metric to measure the goodness of a classifier is the accuracy or classification rate which is defined as the probability of classifying a random selected instance correctly (Kohavi, 1995). Therefore, an accurate estimation of the classification rate is crucial. This estimation is relatively straightforward when the data are plentiful. Nevertheless, accuracy estimation is normally performed with a limited number of cases. Therefore, two main difficulties arise: bias and variance.

- **Bias in the accuracy estimation**. The bias in the accuracy estimation can be seen as the error between the true accuracy of the learned classifier and the accuracy estimated by a specific accuracy estimation method.

The observed accuracy of the learned classifier over the training set is usually a poor estimation of the accuracy for future unseen data. This is specially likely when the rich space over the learning models enables the classifier to overfit the training data. In order to obtain an unbiased accuracy estimation, the classifier has to be evaluated in a set of data samples independent of the data samples used to learn the classifier.

- **Variance in the accuracy estimation**. Even when the accuracy is estimated from data samples that differ from the data samples used to learn the classifier, the estimated accuracy may vary from the true classification accuracy. This is specially critical when the number of test samples used to estimate the accuracy is small. In these situations the estimated accuracy tends to have a large variance.

There are several classical methods to estimate the accuracy of a classifier:

- The **resubstitution error** estimates the accuracy from the same dataset used to learn the classifier. Although the variance of the accuracy estimation is zero, this is not a desirable estimation method because of the high bias; it provides accuracy estimations which are too optimistic.
- The **hold-out estimation** randomly splits the dataset into two parts: training and test sets. Usually, 2/3 of the original dataset are used to learn the classification model (training set) and the remaining 1/3 is used to validate the classifier (test set) and then obtain the accuracy. This accuracy measured in the test set is used as the estimated accuracy for the classifier learned using the whole dataset. An improvement of the hold-out estimation is the random subsampling. This method repeats the hold-out estimation k times and obtains the estimated accuracy as an average over the k runs.
- The ***k*-fold cross-validation** (Stone, 1974) randomly splits the data set into k mutually exclusive folds with approximately the same number of data samples. The process is repeated k times and, at each iteration, $k-1$ folds of the dataset are used to learn the model and the remaining fold is used to calculate the accuracy. The estimated accuracy of the classifier learned from the whole dataset is given by the mean over the k iterations. Additionally, the standard deviation of the averaged value is usually reported. Although the number of folds may vary, a usual choice is $k = 10$. As well as in the hold-out estimation, an improvement for the k-fold cross-validation can be obtained by repeating the cross-validation process several times (Kohavi, 1995). This leads to a lower bias and variance estimations. For instance, Dietterich (1998) propose a 5 times 2-fold cross-validation. Another possible improvement is the stratified cross-validation (Breiman *et al.*, 1984) where folds obtained by splitting the dataset contain the same proportions of classes as the original dataset.
  A particular case of the k-fold cross-validation is the leaving-one-out estimation (Lachenbruch and Michey, 1968). In this method, a $k = N$ is selected and then the cross-validation process is repeated as many times

as samples are in the dataset. Therefore, at each iteration of the cross-validation process the classifier is learned from a dataset of $N-1$ samples and it is evaluated on a single sample. The leaving-one-out method obtains almost unbiased accuracy estimations (Lachenbruch and Michey, 1968).

Although the above exposed methods are, in general, accepted and widely used in the machine learning community, they have also been criticized (Ng, 1997; Provost *et al.*, 1998; Nadeau and Bengio, 2003). Alternatively, other methods that improve the bias and variance of the accuracy estimation, such as jackknife (Rao and Shao, 1992), bootstrap (Efron and Tibshirani, 1993) or bolstered (Braga-Neto and Dougherty, 2004) have been proposed.

3.3 Comparing Learning Algorithms

In a k-fold cross-validation or in other resampling estimation methods, the estimation of the accuracy is given as an average over several accuracy values. When comparing two different learning algorithms, it may happen that even when one algorithm is, with respect to the average of the estimated accuracy, better than the other, there are no real differences between them because of the variance. Several tests have been proposed to measure if the difference between two learning algorithms is statistically significant.

One of the most used tests is the k-fold cross-validation paired t-test (Dietterich, 1998). Let say that A and B are two different algorithms to learn classification models. The data set is split into k disjoint folds of equal size. At the i-th iteration of the cross-validation process, the i-th fold is used as the test set, and the estimated accuracy for the learning algorithms in the i-th fold is represented by random variables A_i and B_i respectively. The test assumes that the random variables A_i and B_i, with $i = 1, \ldots, k$, are normally distributed and the null hypothesis states equal means for the estimated accuracies in both algorithms, $\overline{a} = \frac{1}{k}\sum_{i=1}^{k} a_i$ and $\overline{b} = \frac{1}{k}\sum_{i=1}^{k} b_i$. If we denote as p_i the difference in the estimated accuracy between algorithms A and B for the i-th fold ($p_i = a_i - b_i$, and $\overline{p} = \sum_{i=1}^{k} p_i$) then, the null hypothesis can be tested by using the statistic

$$t = \frac{\overline{p}\sqrt{k}}{\sqrt{\frac{\sum_{i=1}^{k}(p_i-\overline{p})^2}{k-1}}} \tag{3.9}$$

which follows a Student's t distribution with $k-1$ degrees of freedom.

Two main problems arise from the use of this t-test. A minor problem is related to the fact that the estimated accuracy values for A and B have to be paired, that is, the same partition into k-folds has to be used in the cross-validation process of both A and B algorithms so that A_i and B_i are estimated from the same fold. This restriction can be easily overcome by using the following (unpaired) statistic:

$$t_{unpaired} = \frac{(\overline{a} - \overline{b})\sqrt{k}}{\sqrt{\frac{\sum_{i=1}^{k}(a_i - \overline{a})^2 + \sum_{i=1}^{k}(b_i - \overline{b})^2}{k-1}}} \tag{3.10}$$

The other problem is the normality assumption which, quite often, is not really fulfilled. In these cases, where normally distribution can not be ensured, non-parametric tests such as Wilcoxon signed-rank test (Wilcoxon, 1945) or Mann-Whitney U test (Mann and Whitney, 1947) can be used.

The Wilcoxon signed-rank test is a non-parametric alternative to the paired Student's t-test for two related samples (paired test). The estimation of the accuracy for algorithms A and B is identical to the above described one for the t-test. The null hypothesis also states equal means for the probability distributions of the estimated accuracies in both algorithms A and B. The Wilcoxon signed-rank test computes the difference between the observations $p_i = a_i - b_i$ for $i = 1, \ldots, k$ and then obtains the rank of each p_i, R_i, by ordering the absolute value of the differences, $|p_i|$. The Wilcoxon's test statistics are given by:

$$W^+ = \sum_{i=1}^{k} \left(\phi(p_i > 0) R_i + \frac{1}{2} \phi(p_i = 0) R_i \right) \tag{3.11}$$

$$W^- = \sum_{i=1}^{k} \left(\phi(p_i < 0) R_i + \frac{1}{2} \phi(p_i = 0) R_i \right) \tag{3.12}$$

$$W = \min(W^+, W^-) \tag{3.13}$$

where $\phi(X)$ is an indicator function which is equal to 1 if X is true and 0 otherwise. When k is small, the critical values for the statistic, W, are tabulated but as k increases, the statistic tends, under the null hypothesis, towards a normal distribution with $\mu_W = \frac{k(k+1)}{4}$ and $\sigma_W = \sqrt{\frac{k(k+1)(2k+1)}{24}}$.

The Mann-Whitney is also a non-parametric test for assessing whether or not two samples of observations come from the same distribution, but it does not assume matched-pair data. Therefore, the cross-validation process of both algorithms A and B can be performed independently and without maintaining the data partition. The null hypothesis states that the two samples, $A_1, \ldots, A_k$ and $B_1, \ldots, B_k$, are drawn from probability distributions with equal means. The Mann-Whitney test arranges all the accuracy estimations into a single ranked series. That is, it ranks all the accuracy estimations without regard from which algorithm they are obtained. If we define by R_A the sum of the ranks for the accuracy estimations obtained by algorithm A then, the Mann-Whitney statistic is calculated as follows:

$$U = k^2 + \frac{k(k+1)}{2} - R_A \tag{3.14}$$

As well as in the Wilcoxon signed-rank test, when k is small, the critical values for the statistic U are tabulated, but as k increases, the statistic tends, under the null hypothesis, towards the normal distribution with $\mu_U = \frac{k^2}{2}$ and $\sigma_U = \sqrt{\frac{k^2(2k+1)}{12}}$. In practice, with $k > 25$, the normal distribution is often used to calculate the p-value of the test.

Throughout this dissertation we perform statistical tests to compare two different algorithms on the same dataset using non-parametric statistical tests. There are other more complicated and restrictive comparisons between classification algorithms such as the ones presented in Demšar (2006) where methods to compare two classifiers over several datasets or multiple classifiers over one or several datasets are proposed. Although the comparison method between classification algorithms adopted in this dissertation is simpler than the ones introduced in Demšar (2006), we follow the recommendation given in the paper about the fact that non-parametric statistical tests are preferred to parametric ones because they avoid the normality assumption.

4

Bayesian Network Classifiers

Bayesian network classifiers arise from the use of Bayesian network models for classification purposes. Depending on the restrictions imposed on the structure of the Bayesian network, several classifiers have been proposed. However, if no restrictions are assumed, the classification model is given by a general Bayesian network used for classification purposes and it is denoted as unrestricted Bayesian network classifier. Nevertheless, unrestricted Bayesian network classifiers do not necessarily lead to a classifier with better performance. For example, Friedman *et al.* (1997) observed that unrestricted Bayesian network classifiers do not outperform other simpler models.

Bayesian network classifiers are considered generative classifiers because they encode the joint probability distribution of the class and predictive variables. In this case, the maximum likelihood estimation of the parameters is straightforward and it is normally used to learn the parameters of the model. However, several discriminative learning approaches for Bayesian network classifiers have been proposed (Greiner *et al.*, 2005; Roos *et al.*, 2005; Santafé *et al.*, 2005b; Feelders and Ivanovs, 2006). This issue is addressed in the next chapter where the work proposed in Santafé *et al.* (2005b) is presented in deep.

In this chapter, we introduce several restricted Bayesian network classifiers such as naive Bayes, selective naive Bayes, tree augmented naive Bayes, k-dependence Bayesian classifier and Bayesian network augmented naive Bayes.

4.1 Naive Bayes

The naive Bayes (Minsky, 1961) is the simplest Bayesian network classifier. It is based on Bayes' theorem and on the assumption that the predictive variables are conditionally independent given the class variable. Although the naive Bayes model was previously used in statistics and pattern recognition (Duda and Hart, 1973), the first time it appears in supervised classification is in Cestnik *et al.* (1987). In the literature, the naive Bayes paradigm is

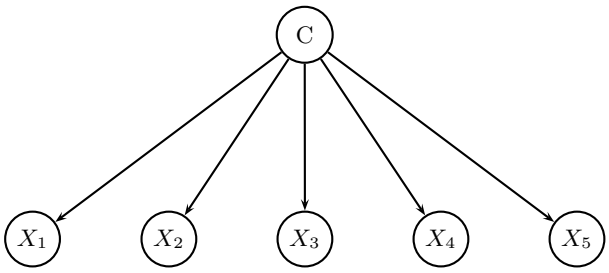


Fig. 4.1. Structure of a naive Bayes model with five predictive variables.

designated with several names: idiot Bayes (Ohmann *et al.*, 1988; Hand and You, 2001), naive Bayes (Kononenko, 1990), simple Bayes (Gammerman and Thatcher, 1991; Domingos and Pazzani, 1997) or independent Bayes (Todd and Stamper, 1994). Figure 4.1 shows the graphical representation of naive Bayes structure with five predictive variables. Additionally, the factorization of the joint probability distribution under the naive Bayes assumption is given by:

$$p(c, \boldsymbol{x}) = p(c) \prod_{i=1}^{n} p(x_i|c) \tag{4.1}$$

In spite of its simplicity, the naive Bayes has been widely used in machine learning applications and, even when naive Bayes assumption is hardly ever fulfilled in real domains, in many cases naive Bayes obtains promising results and is competitive with other more sophisticated methods. For instance, there are successful applications of naive Bayes classifiers in medical domains (Kononenko, 1990; Ohmann *et al.*, 1996; Mani *et al.*, 1997; Movellan *et al.*, 2002) in web site classification according to user interest (Pazzani *et al.*, 1996), in collaborative filter approaches (Miyahara and Pazzani, 2000), text classification (McCallum and Nigam, 1998) or failure detection (Hamerly and Elkan, 2001). For more details about naive Bayes, the reader may be interested on the historical review and improvements of the naive Bayes classifier presented in Larrañaga (2004) and Blanco (2005).

4.2 Selective Naive Bayes

Despite the good performance of naive Bayes classifiers even when the assumption of conditional independence of the variables is not fulfilled, naive Bayes is very sensitive to highly correlated features. The selective naive Bayes (Langley and Sage, 1994) is a variant of the naive Bayes classifier that uses

only a subset of the variables to make predictions (see Figure 4.2 for a graphical example of a selective naive Bayes structure with five predictive variables where only three are selected to make predictions). In fact, the selective naive Bayes classifier can be seen as a naive Bayes classifier for which a feature subset selection process has been performed. This feature subset selection process selects those features or variables which are relevant for classification purposes and these are the variables included in the naive Bayes model.

In general, any feature subset selection method can be used to obtain the subset of relevant and non-redundant variables for classification purposes and thus obtain the selective naive Bayes model. In the literature of feature subset selection two main tendencies appear. On the one hand, filter methods use optimization criteria which are independent of the classification model such as entropy, mutual information, Euclidean distance or Kullback-Leibler divergence (Ben-Bassat, 1982; Doak, 1992; Inza *et al.*, 2004; Blanco *et al.*, 2005). On the other hand, wrapper methods use model-based scores such as accuracy to find the set of relevant variables for classification. These latter methods usually use optimization techniques such as hill-climbing (Langley and Sage, 1994; Inza *et al.*, 2002), simulated annealing (Vinciotti *et al.*, 2006), genetic algorithms (Inza *et al.*, 2001b,c) or estimation of distribution algorithms (Inza *et al.*, 2000, 2001a; Blanco *et al.*, 2004) among others. By contrast, Dash and Cooper (2002), Cerquides and López de Mántaras (2003a) and Dash and Cooper (2004) propose a Bayesian approach by averaging over all the selective naive Bayes models in order to obtain a single naive Bayes model which, despite including all the predictive variables in the model, is able to capture the relevance of the variables for classification purposes.

Langley and Sage (1994) propose a wrapper feature selection process where a greedy forward algorithm is used. The process starts with an empty set of variables and, at each step, the variable which most increases the accuracy measured in a leaving-one-out cross-validation is added to the model. Figure 4.3 shows the pseudo-code of the algorithm proposed in Langley and Sage (1994).

Therefore, the factorization of the joint probability distribution for a selective naive Bayes is given as follows:

$$p(c, \boldsymbol{x}) = p(c) \prod_{X_i \notin \boldsymbol{X}_F} p(x_i) \prod_{X_j \in \boldsymbol{X}_F} p(x_j|c) \tag{4.2}$$

where $\boldsymbol{X}_F$ is the set of selected features to make predictions. Therefore, in the case of selective naive Bayes models, the conditional probability of the class given the predictive variables is given by:

$$p(c|\boldsymbol{x}) = p(c|\boldsymbol{x}_F) \propto p(c, \boldsymbol{x}_F) = p(c) \prod_{X_j \in \boldsymbol{X}_F} p(x_j|c) \tag{4.3}$$

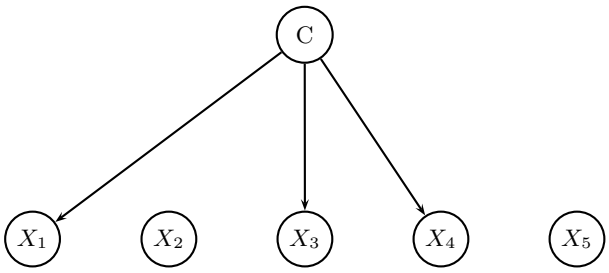


Fig. 4.2. Structure of a selective naive Bayes with five predictive variables where only three are selected to make predictions.

Let the set of selected variables, $\boldsymbol{X}_F$, be empty

While some improvement is reached **do**

  Select the most accurate predictive variable not in $\boldsymbol{X}_F$

  Add the selected variable to $\boldsymbol{X}_F$

end while

Fig. 4.3. General pseudo-code for wrapper approach to selective naive Bayes learning (Langley and Sage, 1994).

4.3 Tree Augmented Naive Bayes

Neither the naive Bayes nor the selective naive Bayes can notice dependencies between predictive variables. The tree augmented naive Bayes (TAN) takes into account relationships between predictive variables by extending the naive Bayes structure with a tree structure among the predictive variables (see Figure 4.4 for a graphical example of a TAN structure with five predictive variables).

The algorithm to learn TAN models was proposed in Friedman *et al.* (1997). In this work Friedman et al. adapt the Chow-Liu algorithm (Chow and Liu, 1968) in order to take into account the special role of the class variable. The construct-TAN algorithm calculates the conditional mutual information between each pair of predictive variables X_i and X_j given the class, with $i, j = 1, \ldots, n$ and $i \neq j$. The conditional mutual information between X_i and X_j given the class variable is defined as:

$$I(X_i, X_j|C) = \sum_{x_i} \sum_{x_j} \sum_{c} p(x_i, x_j, c) \log \frac{p(x_i, x_j|c)}{p(x_i|c)p(x_j|c)} \tag{4.4}$$

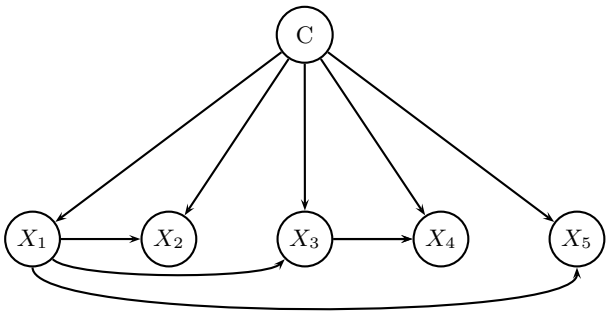


Fig. 4.4. Structure of a TAN classifier with five predictive variables.

Compute $I(X_i, X_j|C)$ for $i < j$ and $i, j = 1, \ldots, n$

Let G be a complete undirected graph where $I(X_i, X_j|C)$ is the weight of edge $X_i - X_j$

Use Kruskal algorithm to obtain the maximum spanning tree from G

Select randomly a node as the root to set the direction of the edges

Add the class variable as parent of each predictive variable

Fig. 4.5. General pseudo-code for TAN classifier (Friedman *et al.*, 1997).

The algorithm needs the calculation of $n(n-1)/2$ mutual information values. Each conditional mutual information value $I(X_i, X_j|C)$ is used to weight the edge between X_i and X_j. Then, the Kruskal algorithm is used to obtain the maximum weighted spanning tree. Figure 4.5 shows the pseudo-code for the construct-TAN algorithm and Figure 4.6 shows the pseudo-code for the Kruskal algorithm.

The construct-TAN algorithm builds a TAN model that maximizes the likelihood and, as well as the Chow-Liu algorithm, it is asymptotically correct if the data have been generated from a TAN structure. That is, if the dataset is large enough, the construct-TAN algorithm is able to learn the original TAN structure.

Keogh and Pazzani (1999) propose a wrapper greedy alternative approach to obtain a TAN structure where, starting with a naive Bayes, the arcs that maximize the proposed score are successively added if they agree with TAN structural restrictions. By contrast, Lucas (2004) introduces the forest augmented naive Bayes (FAN) which relaxes the TAN restrictions by allowing the predictive variables to form up to a tree. That is, a $k < n - 1$ number must be fixed in order to obtain the forest with the minimum cost and with exactly k edges (Figure 4.7 shows a graphical example of a FAN structure).

Let T be an empty graph with n variables

Add to T the two edges with the highest weight

while number of edges in T is less than $n-1$ **do**

Add the edge with the highest weight which does not create a cycle in T

end while

Fig. 4.6. General pseudo-code for Kruskal algorithm.

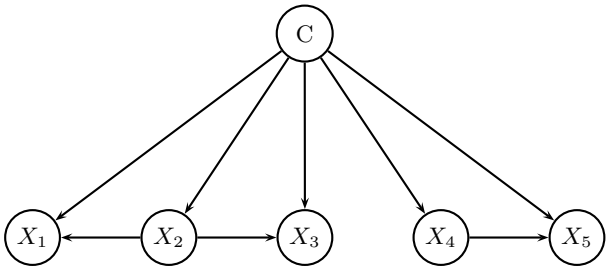


Fig. 4.7. Structure of a FAN classifier with five predictive variables.

Another alternative to TAN construction is proposed in Pernkopf and Bilmes (2005), and Perez *et al.* (2006) where Friedman et al.'s algorithm is modified in order to perform a discriminative learning of TAN structures by maximizing the conditional likelihood instead of the joint likelihood.

Moreover, there are some Bayesian approaches for TAN induction. Dash and Cooper (2003), Cerquides and López de Mántaras (2003a), Dash and Cooper (2004), and Cerquides and López de Mántaras (2005) introduce different tractable approaches to average over all the possible TAN structures given an ancestral order over the predictive variables. However, although the Bayesian model averaging process is performed over TAN models, the resultant model is a complete Bayesian network. Nevertheless, Dash and Cooper (2004) propose some restrictions to make the calculations more efficient.

4.4 *k* Dependence Bayesian Classifier

TAN models restrict the number of parents for a predictive variable to a maximum of two: the class variable and another predictive variable. The k dependence Bayesian classifier (kDB) (Sahami, 1996) relax TAN restrictions by allowing each predictive variable to have up to k predictive variables as parents besides the class variable, k being a parameter of the model which

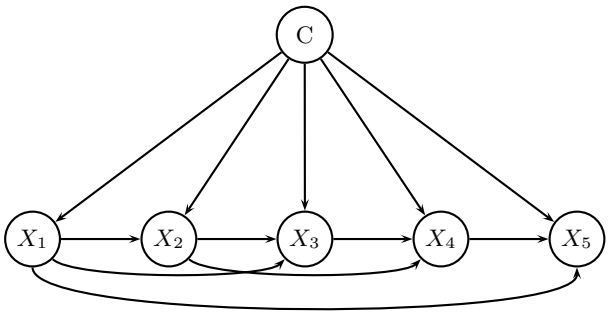


Fig. 4.8. Structure of a kDB classifier with five predictive variables.

Compute $I(X_i, C)$ for $i = 1, \ldots, n$

Compute $I(X_i, X_j|C)$ for $i < j$ and $i, j = 1, \ldots, n$

Let S a structure with only the class variable C

while $|S| \neq n + 1$ **do**

  Select the variable, X_{max}, which is not in S and has the highest $I(X_{max}|C)$

  Add X_{max} and the arc $C \rightarrow X_{max}$ to S

  for $m = 1$ **to** $\min(|S|, k)$ **do**

    Select the variable X_j in S with the highest $I(X_{max}, X_j|C)$

    Add to S the arc $X_j \rightarrow X_{max}$

  end for

end while

Fig. 4.9. General pseudo-code for kDB classifier (Sahami, 1996).

may be fixed in advance. The kDB classifier can be seen as a generalization of other simpler models such as naive Bayes ($k = 0$) or TAN ($k = 1$). Figure 4.8 shows a graphical example of kDB structure with five predictive variables.

However, as kDB classifiers force each predictive variable (except for the first k variables introduced in the model) to have exactly k predictive variables as parents and the value of k is fixed by hand, the classifier may not be optimum. Figure 4.9 shows the pseudo-code for kDB algorithm.

4.5 Bayesian Network Augmented Naive Bayes

The Bayesian network augmented naive Bayes (BAN) is a generalization of the kDB classifier which allows the predictive variables to form any unrestricted Bayesian network and where the class variable is added as parent of each predictive variable. Figure 4.10 shows a graphical representation of a BAN classifier with five predictive variables.

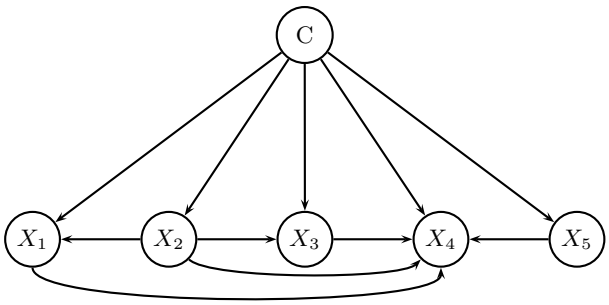


Fig. 4.10. Structure of a BAN classifier with five predictive variables.

Although the term Bayesian network augmented naive Bayes is introduced in Cheng and Greiner (2001), this classifier was previously used and referred with the general term of augmented naive Bayes. In fact, Friedman *et al.* (1997) propose the use of a MDL score metric and a local search algorithm to learn an unrestricted augmented naive Bayes classifier; in Ezawa *et al.* (1996) the model is learned using the K2 algorithm (Cooper and Herskovits, 1992) and in Zhang and Ling (2001) a complexity measure function to learn augmented naive Bayes models is introduced. Additionally, Acid *et al.* (2005) propose a local method to explore the search space of C-RPDAGs (class-focused restricted partial directed acyclic graphs) structures in order to learn the structure of BAN and other unrestricted Bayesian network classifiers.

5

New Approaches for the Discriminative Learning of Bayesian Network Classifiers

The learning of probabilistic classification models can be approached from either a generative or a discriminative point of view. Generative classifiers, also called informative classifiers, obtain the parameters of the model by learning the joint probability distribution, $p(c, \boldsymbol{x})$. These models are obtained by maximizing the joint log-likelihood function. Thus, the classification of a new instance can be performed by selecting the most likely class from $p(c|\boldsymbol{x})$, which can be indirectly obtained by using the Bayes rule. On the other hand, discriminative classifiers obtain the parameters of the model by directly learning the posterior distribution of the class given the predictive variables, $p(c|\boldsymbol{x})$. In this case, the classification model is obtained by maximizing the conditional log-likelihood function.

In this chapter we motivate the use of discriminative learning for Bayesian network classifiers and overview some approaches presented in the literature. The main contribution of this chapter is a novel approach for the discriminative learning of the parameters for Bayesian network classifiers by means of an adaptation of the TM algorithm. Moreover, we also propose two new algorithms (structural TM and construct-discriminative-TAN) which can be used to learn the structure of Bayesian network classifiers from a discriminative point of view.

5.1 Motivation

There is a strong belief in the scientific community that discriminative learning has to be preferred in reasoning tasks. In fact, the selection of a generative or discriminative classifier can be argued in different terms. Vapnik (1998) says, "*... one should solve the (classification) problem directly and never solve a more general problem as an intermediate step (such as modeling $p(c, \boldsymbol{x})$)*". Indeed, generative learning preforms a more general approach by modeling the joint probability distribution, $p(c, \boldsymbol{x})$, and then transforming the joint probability distribution into the conditional model, $p(c|\boldsymbol{x})$, in order to make

predictions. This approach is against the widely applied Occam's razor principle: "*It is futile to do with more what can be done with fewer*" (William of Ockham, 1280-1349). However, the generative approach is widely used to learn some classification models such as Bayesian network classifiers because the decomposability of the log-likelihood (LL) score leads to a simple and efficient estimation of the parameters. By contrast, although discriminative learning seems a more natural approach for classification purposes, the estimation of the parameters is much more inefficient than in generative learning.

Nevertheless, the maximization of the log-likelihood score does not necessarily lead to improving the classification rate (Friedman *et al.*, 1997). In fact, log-likelihood not only depends on the conditional log-likelihood (CLL) but also on the marginal log-likelihood:

$$\begin{aligned} LL &= \sum_{d=1}^{N} \log p(c^{(d)}, \boldsymbol{x}^{(d)}|\mathcal{B}) \\ &= \sum_{d=1}^{N} \log p(c^{(d)}|\boldsymbol{x}^{(d)}, \mathcal{B}) + \sum_{d=1}^{N} \log p(\boldsymbol{x}^{(d)}|\mathcal{B}) \end{aligned} \tag{5.1}$$

As can be seen in Equation 5.1, only the first term (CLL) is related to classification since the second one, marginal likelihood, is only relevant to model the relationship among predictive variables but not for classification purposes. Moreover, as the number of predictive variables increases, the second term in Equation 5.1 should become more relevant than the first one. Therefore, a criteria related to the log-likelihood score might be adequate to learn the (in)dependencies between variables captured in the dataset but could be inadequate to learn a classification model, especially with a high number of predictive variables. By contrast, the conditional log-likelihood score is directly related to the discrimination function $p(c|\boldsymbol{x})$. The maximization of conditional log-likelihood also involves minimizing the entropy of the class given the predictive variables, $H(C|\boldsymbol{X})$ (Perez *et al.*, 2006; Cover and Thomas, 2006). Minimizing $H(C|\boldsymbol{X})$ should be desirable for learning classification models because we minimize the uncertainty remaining in the class variable once the value of $\boldsymbol{X}$ is known.

On the other hand, it is quite usual to restrict the model complexity when learning Bayesian network classifiers. For instance, naive Bayes, TAN and kDB models are Bayesian network classifiers with restricted network complexity. However, the model underlying the dataset (or the model which generated the data) may be more complex than the classifier learned from the dataset. Therefore, the estimation of both conditional log-likelihood (first term in Equation 5.1) and marginal log-likelihood (second term in Equation 5.1) is biased by the complexity restrictions that we set when learning the Bayesian network classifier. Hence, if the assumptions (or the restrictions) in the model that we learn are true, for instance, we learn a TAN classifier and the model

that generated the data is also a TAN model, generative learning may present a good performance since it is able to model the relations between variables. Therefore, as generative learning is computationally more efficient, it may be preferred. In fact, some generative approaches are asymptotically correct. For example, Friedman et al.'s algorithm to learn the structure of a TAN classifier, guarantees that if the model that generated the data is a TAN model, the construct-TAN algorithm, having enough data samples, is able to induce the original TAN structure. By contrast, when the learned model is different from the *true* model, generative learning should perform worse than discriminative learning (Rubinstein and Hastie, 1997) because the bias for the generative model is higher. This can be seen in Theorem 3 presented in this dissertation, but in order to enunciate this theorem, Theorems 1 and 2 are also needed. Theorems 1 and 2 are adaptations of theorems presented in Chow and Liu (1968) and Perez *et al.* (2007) respectively. However, we include the proofs of the theorems for a better understanding.

Theorem 1. *Given a dataset D with N instances and given a model $\mathcal{B} = (S, \boldsymbol{\theta})$, the maximum likelihood parameters are the ones that minimize the Kullback-Leibler divergence between the model and the empirical distribution of the data ($\hat{p}(c, \boldsymbol{x}) = \frac{1}{N}$ iff $(c, \boldsymbol{x}) \in D$ and 0 otherwise).*

Proof.

$$
\begin{aligned}
LL &= N \sum_{d=1}^{N} \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log p(c^{(d)}, \boldsymbol{x}^{(d)} | \mathcal{B}) \\
&= -N \sum_{d=1}^{N} \big(- \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log p(c^{(d)}, \boldsymbol{x}^{(d)} | \mathcal{B}) + \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \\
&\quad - \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \big) \\
&= N \sum_{d=1}^{N} \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) - N \sum_{d=1}^{N} \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log \frac{\hat{p}(c^{(d)}, \boldsymbol{x}^{(d)})}{p(c^{(d)}, \boldsymbol{x}^{(d)} | \mathcal{B})} \\
&= N \sum_{c, \boldsymbol{x}} \hat{p}(c, \boldsymbol{x}) \log \hat{p}(c, \boldsymbol{x}) - N \sum_{c, \boldsymbol{x}} \hat{p}(c, \boldsymbol{x}) \log \frac{\hat{p}(c, \boldsymbol{x})}{p(c, \boldsymbol{x} | \mathcal{B})} \\
&= \kappa - N D_{KL} \big(\hat{p}(c, \boldsymbol{x}) || p(c, \boldsymbol{x} | \mathcal{B}) \big)
\end{aligned}
$$

where κ is a constant. □

Theorem 2. *Given a dataset D with N instances and given a model $\mathcal{B} = (S, \boldsymbol{\theta})$, the parameters of the model that maximize the conditional log-likelihood are the ones that minimize the expected Kullback-Leibler divergence with respect to $\hat{p}(\boldsymbol{x})$ between the conditional model and the empirical conditional distribution of the data ($\hat{p}(c|\boldsymbol{x}) = \frac{\hat{p}(c, \boldsymbol{x})}{\hat{p}(\boldsymbol{x})}$).*

Proof.

$$
\begin{aligned}
CLL &= N \sum_{d=1}^{N} \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log p(c^{(d)}|\boldsymbol{x}^{(d)}, \mathcal{B}) \\
&= -N \sum_{d=1}^{N} \big(- \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log p(c^{(d)}|\boldsymbol{x}^{(d)}, \mathcal{B}) + \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log \hat{p}(c^{(d)}|\boldsymbol{x}^{(d)}) \\
&\quad - \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log \hat{p}(c^{(d)}|\boldsymbol{x}^{(d)}) \big) \\
&= N \sum_{d=1}^{N} \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log \hat{p}(c^{(d)}|\boldsymbol{x}^{(d)}) - N \sum_{d=1}^{N} \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log \frac{\hat{p}(c^{(d)}|\boldsymbol{x}^{(d)})}{p(c^{(d)}|\boldsymbol{x}^{(d)}, \mathcal{B})} \\
&= N \sum_{c, \boldsymbol{x}} \hat{p}(c, \boldsymbol{x}) \log \hat{p}(c|\boldsymbol{x}) - N \sum_{c, \boldsymbol{x}} \hat{p}(c, \boldsymbol{x}) \log \frac{\hat{p}(c|\boldsymbol{x})}{p(c|\boldsymbol{x}, \mathcal{B})} \\
&= \kappa - N E_{\hat{p}(\boldsymbol{x})} \Big[D_{KL} \big(\hat{p}(c|\boldsymbol{x}) || p(c|\boldsymbol{x}, \boldsymbol{\theta}) \big) \Big]
\end{aligned}
$$

where κ is a constant. □

Theorem 3. *Given a dataset D with N instances and given a model $\mathcal{B} = (S, \boldsymbol{\theta})$, if the structure of the model, S, is able to capture all the conditional independencies among the predictive variables in D, the generative learning of the parameters is asymptotically equivalent to the discriminative learning.*

Proof.

$$
\begin{aligned}
D_{KL} \big(\hat{p}(c, \boldsymbol{x}) || p(c, \boldsymbol{x}|\mathcal{B}) \big) &= \sum_{c, \boldsymbol{x}} \hat{p}(c, \boldsymbol{x}) \log \frac{\hat{p}(c, \boldsymbol{x})}{p(c, \boldsymbol{x}|\mathcal{B})} && (5.2) \\
&= \sum_{d=1}^{N} \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \log \frac{\hat{p}(c^{(d)}, \boldsymbol{x}^{(d)})}{p(c^{(d)}, \boldsymbol{x}^{(d)}|\mathcal{B})} && (5.3) \\
&= \sum_{d=1}^{N} \hat{p}(c^{(d)}, \boldsymbol{x}^{(d)}) \left(\log \frac{\hat{p}(c^{(d)}|\boldsymbol{x}^{(d)})}{p(c^{(d)}|\boldsymbol{x}^{(d)}, \mathcal{B})} + \log \frac{\hat{p}(\boldsymbol{x}^{(d)})}{p(\boldsymbol{x}^{(d)}|\mathcal{B})} \right) && (5.4) \\
&= \sum_{d=1}^{N} \hat{p}(c, \boldsymbol{x}) \log \frac{\hat{p}(c|\boldsymbol{x})}{p(c|\boldsymbol{x}, \mathcal{B})} + \sum_{d=1}^{N} \hat{p}(c, \boldsymbol{x}) \log \frac{\hat{p}(\boldsymbol{x})}{p(\boldsymbol{x}|\mathcal{B})} && (5.5)
\end{aligned}
$$

From Theorem 1 we know that Equation 5.3 is related to the log-likelihood score (generative learning), and from Theorem 2 we know that the first term in Equation 5.5 is related to the conditional log-likelihood score (discriminative learning). If the the structure S captures all the conditional independencies represented in the empirical distribution $\hat{p}(\boldsymbol{x})$, given enough data samples, we can find the set of parameters for $\mathcal{B}$ so that the second term in Equation 5.5 is canceled out. Therefore, generative and discriminative learning of the parameters would be equivalent. However, if the structure of $\mathcal{B}$ and the structure of the empirical model are different, the second term in Equation 5.5 will be always greater than zero. Therefore, in that case, the bias of the generative learning is higher. □

As was mentioned before, Bayesian network classifiers are usually considered generative classifiers because they are learned by maximizing the joint likelihood (or some function related to the joint likelihood). However, there has been a considerable growth of interest in the discriminative learning of Bayesian network classifiers. Although Minka (2005) states that the term discriminative learning is a misnomer, most of the authors use it to refer to the learning process of Bayesian network classifiers where the conditional likelihood score is maximized. Minka (2005) justifies his statement with the fact that the discriminative learning of a Bayesian network classifier is actually learning a different model from the one obtained if a generative learning is performed and then it can not be considered discriminative learning but the learning of a discriminative model. Nonetheless, we use both terms, discriminative learning and discriminative model, throughout the dissertation.

In following sections, we present the discriminative learning of both parameters and structure for Bayesian network classifiers. Although it is necessary to set a structure before learning the parameters of a Bayesian network classifier, some aspects of the discriminative parameter learning are used for the learning of structures. Therefore, the parameter learning is presented before the structure learning.

5.2 Discriminative Learning of Parameters for Bayesian Network Classifiers

In the literature there are several proposals for the discriminative learning of the parameters for Bayesian network classifiers. Huang *et al.* (2003, 2005) propose a discriminative approach to learn the parameters of naive Bayes and TAN models by using an optimization function which includes a penalty term describing the divergence between the classes. Roos *et al.* (2005) prove that Bayesian networks satisfying certain graph-theoretic conditions have an equivalent logistic regression model and therefore the conditional log-likelihood has no local maximum for the parameters of those logistic regression models (McLachlan and Krishnan, 1997). Hence, the parameters that maximize the conditional log-likelihood can be, in principle, easily obtained by any simple local optimization method although it is computationally more demanding than using the parameters that maximize the joint likelihood. Similarly, Greiner *et al.* (2005) extend simple logistic regression models to represent any arbitrary Bayesian network structure. However, learning the parameters that maximize the conditional log-likelihood is NP-Hard. Therefore, they propose a gradient descent algorithm that attempts to maximize the conditional log-likelihood by using the extended logistic regression model. Feelders and Ivanovs (2006) propose an alternative mapping between Bayesian network classifiers and logistic regression models similar to the one introduced by Roos *et al.* (2005) but with less parameters in the corresponding logistic regression

model. Finally, Santafé *et al.* (2005b) introduce a novel approach for the discriminative learning of the parameters of Bayesian network classifiers based on the TM algorithm. This work is presented in detail in following sections.

5.2.1 The TM Algorithm

The TM algorithm is a general iterative process that allows the maximization of the conditional log-likelihood in models where the (joint) log-likelihood function is easier to maximize. The algorithm was originally proposed in Edwards and Lauritzen (2001) and the convergence properties were studied in Sundberg (2002). The TM algorithm resembles the well-known EM algorithm (Dempster *et al.*, 1977) by alternating between two steps. The M step maximizes a function related to the conditional log-likelihood function and the T step computes the conditional expectation to identify this function. However, it differs from the EM algorithm and its variants by being applied to the complete dataset and by augmenting the parameters rather than the data. In the following sections we firstly introduce the TM algorithm in the same way as in Edwards and Lauritzen (2001) but bearing in mind the classification purpose of the model that we want to learn. Thus, we expect to give the reader a general and intuitive idea about how the TM algorithm works. Then, we introduce the adaptation of the TM algorithm for learning the parameters of Bayesian network classifiers (Santafé *et al.*, 2004, 2005b).

5.2.1.1 General Structure of the TM Algorithm

For the sake of clarity, let us re-name the joint, marginal and conditional log-likelihood functions as follows:

$$\begin{aligned} l(\boldsymbol{\theta}) &= \log f(c, \boldsymbol{x}|\boldsymbol{\theta}) \\ l_{\boldsymbol{x}}(\boldsymbol{\theta}) &= \log f(\boldsymbol{x}|\boldsymbol{\theta}) \\ l^{\boldsymbol{x}}(\boldsymbol{\theta}) &= \log f(c|\boldsymbol{x}, \boldsymbol{\theta}) \end{aligned} \tag{5.6}$$

where $\boldsymbol{\theta}$ is the parameter set of the joint probability distribution for the variable $(C, \boldsymbol{X})$.

The foundations of the TM algorithm are based on the tilted joint log-likelihood function, $q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$. This function is an approximation to $l^{\boldsymbol{x}}(\boldsymbol{\theta})$, which we want to maximize, at a point $\boldsymbol{\theta}^{(t)}$. Note that $l^{\boldsymbol{x}}(\boldsymbol{\theta})$ can be expressed in terms of the joint and the marginal log-likelihood:

$$l^{\boldsymbol{x}}(\boldsymbol{\theta}) = l(\boldsymbol{\theta}) - l_{\boldsymbol{x}}(\boldsymbol{\theta}) \tag{5.7}$$

Therefore, if we expand $l_{\boldsymbol{x}}(\boldsymbol{\theta})$ in a first order Taylor series about the point $\boldsymbol{\theta}^{(t)}$, and then omit the terms which are constant with respect to $\boldsymbol{\theta}$, we can approximate $l^{\boldsymbol{x}}(\boldsymbol{\theta})$ by $q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ as follows:

$$l^{\boldsymbol{x}}(\boldsymbol{\theta}) \approx q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = l(\boldsymbol{\theta}) - \boldsymbol{\theta}^T \dot{l}_{\boldsymbol{x}}(\boldsymbol{\theta}^{(t)}) \tag{5.8}$$

where $\dot{l}_{\boldsymbol{x}}(\boldsymbol{\theta}^{(t)})$ is the derivative of $l_{\boldsymbol{x}}(\boldsymbol{\theta})$ at point $\boldsymbol{\theta}^{(t)}$.

The tilted joint log-likelihood and the conditional log-likelihood functions have the same gradient at $\boldsymbol{\theta}^{(t)}$. Thus, we can maximize $l^{\boldsymbol{x}}(\boldsymbol{\theta})$ by maximizing $q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$.

Since the approximation of $l^{\boldsymbol{x}}(\boldsymbol{\theta})$ given by $q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ is at point a $\boldsymbol{\theta}^{(t)}$, an iterative process is needed in order to maximize the conditional log-likelihood function. This process alternates between two steps, T and M. In the T step, the above described tilted joint log-likelihood is obtained. The second step of the algorithm, the M step, consists of maximizing the tilted joint log-likelihood function:

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \quad q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) \tag{5.9}$$

Under regularity conditions of the usual type and due to the fact that the expected score statistic for the conditional model is equal to 0, $\dot{l}_{\boldsymbol{x}}(\boldsymbol{\theta})$ can be calculated as the expectation of the score statistic for the joint model:

$$\dot{l}_{\boldsymbol{x}}(\boldsymbol{\theta}) = E_{\boldsymbol{\theta}}\{\dot{l}_{\boldsymbol{x}}(\boldsymbol{\theta})|\boldsymbol{x}\} = E_{\boldsymbol{\theta}}\{\dot{l}(\boldsymbol{\theta}) - \dot{l}^{\boldsymbol{x}}(\boldsymbol{\theta})|\boldsymbol{x}\} = E_{\boldsymbol{\theta}}\{\dot{l}(\boldsymbol{\theta})|\boldsymbol{x}\} \tag{5.10}$$

Therefore, the M step involves the solution of the following equation:

$$E_{\boldsymbol{\theta}^{(t)}}\{\dot{l}(\boldsymbol{\theta}^{(t)})|\boldsymbol{x}\} = \dot{l}(\boldsymbol{\theta}) \tag{5.11}$$

In summary, the relevance of the TM algorithm is that it allows to obtain a model that maximizes the conditional log-likelihood, $l^{\boldsymbol{x}}(\boldsymbol{\theta})$, by using the joint log-likelihood, $l(\boldsymbol{\theta})$. This is very useful for models such as Bayesian network classifiers, where the obtention of the joint (generative) model is much easier than the obtention of the conditional (discriminative) one.

The TM algorithm begins by making its initial parameters the ones which maximize the joint log-likelihood given the dataset. Then, both the T and the M steps are repeated until the value of the conditional log-likelihood converges. Convergence properties for the TM algorithm are given in Sundberg (2002).

5.2.1.2 The TM Algorithm for the Exponential Family

The TM algorithm can be easily particularized for probability distributions belonging to the exponential family. In this case, the joint and conditional log-likelihood are given by the following expressions:

$$l(\boldsymbol{\theta}) = \boldsymbol{\alpha}^T u(c, \boldsymbol{x}) + \boldsymbol{\beta}^T v(\boldsymbol{x}) - \psi(\boldsymbol{\alpha}, \boldsymbol{\beta}) \tag{5.12}$$

$$l^x(\boldsymbol{\theta}) = \boldsymbol{\alpha}^T u(c, \boldsymbol{x}) - \psi^x(\boldsymbol{\alpha}) \tag{5.13}$$

with

$$\psi(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \log \int exp\{\boldsymbol{\alpha}^T u(c, \boldsymbol{x}) + \boldsymbol{\beta}^T v(\boldsymbol{x})\}\mu(dc|\boldsymbol{x})\mu(d\boldsymbol{x})$$
$$\psi^x(\boldsymbol{\alpha}) = \log \int exp\{\boldsymbol{\alpha}^T u(c, \boldsymbol{x})\}\mu(dc|\boldsymbol{x}) \tag{5.14}$$

where $\boldsymbol{\alpha}$ denotes the parameters of the conditional model and $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ represent the parameters of the joint model. Let us introduce a new parametrization for $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\eta})$, with:

$$\boldsymbol{\eta} = \frac{\partial}{\partial \boldsymbol{\beta}}\psi(\boldsymbol{\alpha}, \boldsymbol{\beta}) \tag{5.15}$$

Moreover, if we define two new random variables, $\boldsymbol{\mathcal{U}} = u(C, \boldsymbol{X})$ and $\boldsymbol{\mathcal{V}} = v(\boldsymbol{X})$, it can be demonstrated that the maximum likelihood parameters are $\hat{\boldsymbol{\theta}} = (\boldsymbol{u}, \boldsymbol{v})$ with $\boldsymbol{u} = E_{\boldsymbol{\theta}}\{\boldsymbol{\mathcal{U}}\}$ and $\boldsymbol{v} = \boldsymbol{\eta} = E_{\boldsymbol{\theta}}\{\boldsymbol{\mathcal{V}}\}$.

Following the general structure of the TM algorithm, Equation 5.11 has to be solved in order to maximize the approximation to the conditional log-likelihood given by $q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$. Thus, we have:

$$\begin{aligned} E_{\boldsymbol{\theta}}\left\{\frac{\partial}{\partial \boldsymbol{\theta}} l(\boldsymbol{\theta}) \middle| \boldsymbol{x}\right\} &= E_{\boldsymbol{\theta}}\left\{\boldsymbol{\mathcal{U}} - \frac{\partial}{\partial \boldsymbol{\alpha}}\,\psi(\boldsymbol{\alpha}, \boldsymbol{\beta}), \boldsymbol{\mathcal{V}}^T\,\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\eta}} - \frac{\partial}{\partial \boldsymbol{\beta}}\psi(\boldsymbol{\alpha}, \boldsymbol{\beta})\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\eta}} \middle| \boldsymbol{x}\right\} \\ &= \left(E_{\boldsymbol{\theta}}\{\boldsymbol{\mathcal{U}}|\boldsymbol{x}\} - E_{\boldsymbol{\theta}}\{\boldsymbol{\mathcal{U}}\}, (E_{\boldsymbol{\theta}}\{\boldsymbol{\mathcal{V}}\} - \boldsymbol{\eta})^T\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\eta}}\right) \\ &= (E_{\boldsymbol{\theta}}\{\boldsymbol{\mathcal{U}}|\boldsymbol{x}\} - E_{\boldsymbol{\theta}}\{\boldsymbol{\mathcal{U}}\}, 0) \end{aligned} \tag{5.16}$$

and also:

$$\begin{aligned} \dot{l}(\boldsymbol{\theta}) &= \left(\boldsymbol{\mathcal{U}} - \frac{\partial}{\partial \boldsymbol{\alpha}}\psi(\boldsymbol{\alpha}, \boldsymbol{\eta}), \boldsymbol{\mathcal{V}}^T\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\eta}} - \frac{\partial}{\partial \boldsymbol{\beta}}\psi(\boldsymbol{\alpha}, \boldsymbol{\beta})\frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\eta}}\right) \\ &= (\boldsymbol{\mathcal{U}} - E_{\boldsymbol{\theta}}\{\boldsymbol{\mathcal{U}}\}, 0) \end{aligned} \tag{5.17}$$

Finally, the solution of Equation 5.11 gives the value of the sufficient statistics at the $t + 1$-th iteration of the TM algorithm:

$$\begin{aligned} \boldsymbol{u}^{(r+1)} &= \boldsymbol{u}^{(t)} + \boldsymbol{u}^{(0)} - E_{\boldsymbol{\theta}^{(t)}}\{\boldsymbol{\mathcal{U}}|\boldsymbol{x}\} \\ \boldsymbol{\theta}^{(t+1)} &= \hat{\boldsymbol{\theta}}(\boldsymbol{u}^{(t+1)}, \boldsymbol{v}) \end{aligned} \tag{5.18}$$

where the initial sufficient statistics, $\boldsymbol{u}^{(0)}$ and $\boldsymbol{v}$, are given by the maximum likelihood estimators obtained from the data set. Moreover, $\hat{\boldsymbol{\theta}}(\boldsymbol{u}^{(t+1)}, \boldsymbol{v})$ denotes the maximum likelihood estimations of $\boldsymbol{\theta}$ obtained from sufficient statistics $\boldsymbol{u}^{(t+1)}$ and $\boldsymbol{v}$.

Generally, it may happen that an iteration of the TM algorithm yields an illegal set of parameters $\boldsymbol{\theta}$ or that the conditional log-likelihood decreases from one iteration to another. These situations must be corrected by applying a linear search. Thus, the sufficient statistics at iteration $t+1$ are calculated as:

$$\boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t)} + \lambda(\boldsymbol{u}^{(0)} - E_{\boldsymbol{\theta}^{(t)}}\{\boldsymbol{\mathcal{U}}|\boldsymbol{x}\}), \quad \text{with } \lambda \in (0,1) \tag{5.19}$$

being λ the one that maximizes the conditional log-likelihood.

5.2.2 The TM Algorithm for Bayesian Classifiers

In this section we present the work introduced in Santafé *et al.* (2004) and Santafé *et al.* (2005b) where the TM algorithm is adapted to Bayesian network classifiers such as naive Bayes, FAN and BAN. Even when Bayesian networks belong to the exponential family, the adaptation of the calculations from Section 5.2.1.2 is not trivial. First, we introduce the TM algorithm for the discriminative learning of naive Bayes, FAN and BAN models with dichotomic variables because the calculations are simpler and these derivations may lead to a better understanding of the algorithm. Then, the TM algorithm is adapted for the discriminative learning of naive Bayes and FAN classifiers with multinomial variables. Since TAN can be seen as a special case of FAN models, the derivation for TAN models can be straightforward obtained from the TM algorithm for FAN models and therefore, these calculations are not presented in the dissertation.

5.2.2.1 The TM Algorithm for Naive Bayes Models with Dichotomic Variables

We assume that each predictive variable, X_i, and C can take binary values, and we represent the values of C as c and $\overline{c}$ respectively.

The general algorithm for probability distributions of the exponential family requires the expression of the joint log-likelihood via Equation 5.12. This can be achieved by using the naive Bayes conditional model:

$$\begin{aligned} p(c|\boldsymbol{x}) &= \frac{p(c,\boldsymbol{x})}{p(\boldsymbol{x})} = \frac{p(c)\prod_{i=1}^{n} p(x_i|c)}{p(\boldsymbol{x})} \\ &= \frac{p(c)\prod_{i=1}^{n} p(x_i|c)}{p(c)\prod_{i=1}^{n} p(x_i|c) + p(\overline{c})\prod_{i=1}^{n} p(x_i|\overline{c})} \end{aligned} \tag{5.20}$$

Plugging the conditional model into the joint model, we obtain the following expression:

$$p(c, \boldsymbol{x}) = \frac{1}{(p(c))^{n-1}} \prod_{i=1}^{n} p(x_i, c) \tag{5.21}$$

If only one data sample is taken into account, the joint likelihood can be written as:

$$\begin{aligned} p(c, \boldsymbol{x}) &= (p(C=0)^{(1-c)}(1-p(C=0))^{c})^{-(n-1)} \\ &\quad \prod_{i=1}^{n} \Big[p(C=0, X_i=0)^{(1-c)(1-x_i)} p(C=0, X_i=1)^{(1-c)x_i} \\ &\quad p(C=1, X_i=0)^{c(1-x_i)} p(C=1, X_i=1)^{cx_i} \Big] \\ &= \exp\Big\{ \log\Big((p(C=0)^{(1-c)}(1-p(C=0))^{c})^{-(n-1)} \\ &\quad \prod_{i=1}^{n} \Big[p(C=0, X_i=0)^{(1-c)(1-x_i)} p(C=0, X_i=1)^{(1-c)x_i} \\ &\quad p(C=1, X_i=0)^{c(1-x_i)} p(C=1, X_i=1)^{cx_i} \Big] \Big) \Big\} \end{aligned} \tag{5.22}$$

Then, we can rewrite Equation 5.22 as Equation 5.12 by sorting the terms in the equation:

$$\begin{aligned} p(c, \boldsymbol{x}) &= \exp\{ c\,((n-1)(\log(p(C=0)) - \log(1-p(C=0)))) \\ &\quad + \sum_{i=1}^{n} cx_i \left(\log(p(C=0, X_i=0)) \log(p(C=1, X_i=1))\right. \\ &\quad - \log(p(C=0, X_i=1)) - \log(p(C=1, X_i=0))) \\ &\quad + \sum_{i=1}^{n} x_i \left(\log(p(C=0, X_i=1)) - \log(p(C=0, X_i=0))\right) \\ &\quad + c \sum_{i=1}^{n} \log p(C=1, X_i=0) - \log p(C=0, X_i=1) \\ &\quad + \sum_{i=1}^{n} \log(p(C=0, X_i=0)) - (n-1)\log(p(C=0))\,\} \end{aligned} \tag{5.23}$$

If we have a dataset with N samples then, the joint log-likelihood of the data given the parameters $\boldsymbol{\theta}$ is given by:

$$
\begin{aligned}
l(\boldsymbol{\theta}) = & \sum_{d=1}^{N} c^{(d)} \left((n-1)(\log(p(C=0)) - \log(1-p(C=0)))\right) \\
& + \sum_{d=1}^{N}\sum_{i=1}^{n} c^{(d)} x_i^{(d)} \left(\log(p(C=0, X_i=0))\log(p(C=1, X_i=1))\right. \\
& - \left.\log(p(C=0, X_i=1)) - \log(p(C=1, X_i=0))\right) \\
& + \sum_{d=1}^{N}\sum_{i=1}^{n} x_i^{(d)} \left(\log(p(C=0, X_i=1)) - \log(p(C=0, X_i=0))\right) \\
& + Nc \sum_{i=1}^{n} \log p(C=1, X_i=0) - \log p(C=0, X_i=1) \\
& + N(\sum_{i=1}^{n} \log(p(C=0, X_i=0)) - (n-1)\log(p(C=0))) \qquad (5.24)
\end{aligned}
$$

where $c^{(d)}$ and $x_i^{(d)}$ are the values of C and X_i given by the d-th data sample respectively.

A few transformations in Equation 5.24 can match its terms with those from Equation 5.12. Thus, we obtain:

$$\boldsymbol{\mathcal{U}} = u(C, \boldsymbol{X}) = \left(\sum_{d=1}^{N} c^{(d)}, \sum_{d=1}^{N} c^{(d)} x_1^{(d)}, \ldots, \sum_{d=1}^{N} c^{(d)} x_n^{(d)}\right) \qquad (5.25)$$

$$\boldsymbol{\mathcal{V}} = v(\boldsymbol{x}) = \left(\sum_{d=1}^{N} x_1^{(d)}, \ldots, \sum_{d=1}^{N} x_n^{(d)}\right) \qquad (5.26)$$

and therefore, the sufficient statistics are given by:

$$\boldsymbol{\mathcal{U}} = (N_0, N_{01}, \ldots, N_{0i}, \ldots, N_{0n}) \qquad (5.27)$$

$$\boldsymbol{\mathcal{V}} = (N_1, \ldots, N_i, \ldots, N_n) \qquad (5.28)$$

where

$$N_0 = \#\{C=1\} = \sum_{d=1}^{N} c^{(d)} \qquad (5.29)$$

$$N_{0i} = \#\{C=1, X_i=1\} = \sum_{d=1}^{N} c^{(d)} x_i^{(d)} \quad i = 1, \ldots, n \qquad (5.30)$$

$$N_i = \#\{X_i=1\} = \sum_{d=1}^{N} x_i^{(d)} \quad i = 1, \ldots, n \qquad (5.31)$$

Note that the subindex 0 refers to the class variable.

The choice of the above described sufficient statistics makes the calculations of the TM algorithm simpler. For instance, at $t+1$-th iteration, the calculation of $E[\mathcal{U}|\boldsymbol{x}]$ is given by:

$$E_{\boldsymbol{\theta}^{(t)}}[N_0|\boldsymbol{x}] = \sum_{d=1}^{N} p(C=1|\boldsymbol{X}=\boldsymbol{x}^{(d)}, \boldsymbol{\theta}^{(t)}) \tag{5.32}$$

$$E_{\boldsymbol{\theta}^{(t)}}[N_{0i}|\boldsymbol{x}] = \sum_{d=1}^{N} p(C=1|\boldsymbol{X}=\boldsymbol{x}^{(d)}, \boldsymbol{\theta}^{(t)}) x_i^{(d)} \quad i=1,\ldots,n \tag{5.33}$$

The obtention of $p(C=1|\boldsymbol{X}=\boldsymbol{x}^{(d)}, \boldsymbol{\theta}^{(t)})$, taking into account Equation 5.20, requires the calculation of $p(C=1)$ and $p(X_i=1|C=c)$. These can be computed, at each iteration of the algorithm, by using the sufficient statistics:

$$p(C=1) = \frac{N_0^{(t)}}{N} \tag{5.34}$$

$$p(X_i=1|C=1) = \frac{N_{0i}^{(t)}}{N_0^{(t)}} \tag{5.35}$$

$$p(X_i=1|C=0) = \frac{N_i - N_{0i}^{(t)}}{N - N_0^{(t)}} \tag{5.36}$$

where $N_0^{(t)}$ and $N_{0i}^{(t)}$ denote the values of the sufficient statistics N_0 and N_{0i} at the t-th iteration of the algorithm, respectively.

Finally, the t-th iteration of the algorithm involves the following calculations:

- T step:
  - Use $N_0^{(t)}$ and $N_{0i}^{(t)}$ to calculate $p(C=1)$ y $p(X_i=1|C=c)$ with $i=1,\ldots,n$.
  - Obtain the expected values of the sufficient statistics: $E_{\boldsymbol{\theta}^{(t)}}[N_0|\boldsymbol{x}]$ and $E_{\boldsymbol{\theta}^{(t)}}[N_{0i}|\boldsymbol{x}]$ with $i=1,\ldots,n$.
- M step:
  - Compute the new values for the sufficient statistics by using:

$$\boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t)} + \boldsymbol{u}^{(0)} - E_{\boldsymbol{\theta}^{(t)}}[\boldsymbol{\mathcal{U}}|\boldsymbol{x}] \tag{5.37}$$

    where $\boldsymbol{u}^{(0)}$ are the maximum likelihood values for sufficient statistics computed from the dataset and $\boldsymbol{u}^{(t)} = (N_0^{(t)}, N_{01}^{(t)}, \ldots, N_{0n}^{(t)})$
  - Calculate the values for the parameters of the model, $\boldsymbol{\theta}^{(t+1)}$, by using the sufficient statistics $\boldsymbol{u}^{(t+1)}$.

Obtain $\boldsymbol{u}^{(0)}$ from the dataset

while stopping criterion is not met **do**

Calculate $E_{\boldsymbol{\theta}^{(t)}}\{\boldsymbol{\mathcal{U}}|\boldsymbol{x}\}$

Update $\boldsymbol{u}$:

$\boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t)} + \boldsymbol{u}^{(0)} - E_{\boldsymbol{\theta}^{(t)}}\{\boldsymbol{\mathcal{U}}|\boldsymbol{x}\}$

Calculate $\boldsymbol{\theta}^{(t+1)}$ from $\boldsymbol{u}^{(t+1)}$

if illegal $\boldsymbol{\theta}^{(t+1)}$ or conditional log-likelihood decreases **then**

Find the best legal $\boldsymbol{\theta}^{(t+1)}$ via linear search

end if

end while

Fig. 5.1. General pseudo-code for the discriminative learning of Bayesian classifiers with dichotomic variables.

Note that, if the M step results in an illegal set of parameters, we should apply the local search pointed out in Equation 5.19. Figure 5.1 shows a general pseudo-code of the TM algorithm for the discriminative learning of Bayesian network models with dichotomic variables.

5.2.2.2 The TM Algorithm for FAN Models with Dichotomic Variables

In the case of FAN models, we need to differentiate between two kinds of predictive variables: the roots of the trees formed by the predictive variables (these root variables have only one parent, the class variable) and rest of the predictive variables which have two parents, the class and another predictive variable.
We assume, without lost of generality, that the first s variables have only the class variable as parent and the parent set for the reminder $n - s$ variables contains two variables: the class and another predictive variable, $\boldsymbol{Pa}_i = \{C, X_{j(i)}\}$ with $i = s+1, \ldots, n$.

As well as in the TM algorithm for naive Bayes models with dichotomic variables, we should define the conditional model for FAN structures, and use this conditional model into the joint model in order to re-write the log-likelihood in the terms expressed in Equation 5.12. The conditional FAN model is given by:

$$p(c|\boldsymbol{x}) = \frac{p(c,\boldsymbol{x})}{p(\boldsymbol{x})} = \frac{p(c)\prod_{i=1}^{s} p(x_i|c)\prod_{i=s+1}^{n} p(x_i|x_{j(i)},c)}{p(\boldsymbol{x})} \tag{5.38}$$

$$= \frac{p(c)\prod_{i=1}^{s} p(x_i|c)\prod_{i=s+1}^{n} p(x_i|x_{j(i)},c)}{p(c)\prod_{i=1}^{s} p(x_i|c)\prod_{i=s+1}^{n} p(x_i|x_{j(i)},c) + p(\overline{c})\prod_{i=1}^{s} p(x_i|\overline{c})\prod_{i=s+1}^{n} p(x_i|x_{j(i)},\overline{c})}$$

The joint model where we have plugged the conditional model into is:

$$p(c,\boldsymbol{x}) = \frac{\prod_{i=1}^{s} p(x_i,c)\prod_{i=s+1}^{n} p(x_i,x_{j(i)},c)}{p(c)^{s-1}\prod_{i=s+1}^{n} p(x_{j(i)},c)} \tag{5.39}$$

By a mathematical transformation of the log-likelihood for a dataset with N samples we can obtain the following expression:

$$
\begin{aligned}
l(\boldsymbol{\theta}) = & \sum_{d=1}^{N}(1-s)(c^{(d)}\log(p(C=1)) + (1-c^{(d)})\log(p(C=0))) \\
& + \sum_{d=1}^{N}\sum_{i=1}^{s}(1-x_i^{(d)})(1-c^{(d)})\log(p(X_i=0,C=0)) \\
& + \sum_{d=1}^{N}\sum_{i=1}^{s}x_i^{(d)}(1-c^{(d)})\log(p(X_i=1,C=0)) \\
& + \sum_{d=1}^{N}\sum_{i=1}^{s}(1-x_i^{(d)})c^{(d)}\log(p(X_i=0,C=1)) \\
& + \sum_{d=1}^{N}\sum_{i=1}^{s}x_i^{(d)}c^{(d)}\log(p(X_i=1,C=1)) \\
& - \sum_{d=1}^{N}\sum_{i=s+1}^{n}(1-x_{j(i)}^{(d)})(1-c^{(d)})\log(p(X_{j(i)}=0,C=0)) \\
& - \sum_{d=1}^{N}\sum_{i=s+1}^{n}x_{j(i)}^{(d)}(1-c^{(d)})\log(p(X_{j(i)}=1,C=0)) \\
& - \sum_{d=1}^{N}\sum_{i=s+1}^{n}(1-x_{j(i)}^{(d)})c^{(d)}\log(p(X_{j(i)}=0,C=1)) \\
& - \sum_{d=1}^{N}\sum_{i=s+1}^{n}x_{j(i)}^{(d)}c^{(d)}\log(p(X_{j(i)}=1,C=1)) \\
& + \sum_{d=1}^{N}\sum_{i=s+1}^{n}(1-x_i^{(d)})(1-x_{j(i)}^{(d)})(1-c^{(d)})\log(p(X_i=0,X_{j(i)}=0,C=0)) \\
& + \sum_{d=1}^{N}\sum_{i=s+1}^{n}(1-x_i^{(d)})(1-x_{j(i)}^{(d)})c^{(d)}\log(p(X_i=0,X_{j(i)}=0,C=1)) \\
& + \sum_{d=1}^{N}\sum_{i=s+1}^{n}(1-x_i^{(d)})x_{j(i)}^{(d)}(1-c^{(d)})\log(p(X_i=0,X_{j(i)}=1,C=0)) \\
& + \sum_{d=1}^{N}\sum_{i=s+1}^{n}(1-x_i^{(d)})x_{j(i)}^{(d)}c^{(d)}\log(p(X_i=0,X_{j(i)}=1,C=1)) \\
& + \sum_{d=1}^{N}\sum_{i=s+1}^{n}x_i^{(d)}(1-x_{j(i)}^{(d)})(1-c^{(d)})\log(p(X_i=1,X_{j(i)}=0,C=0)) \\
& + \sum_{d=1}^{N}\sum_{i=s+1}^{n}x_i^{(d)}(1-x_{j(i)}^{(d)})c^{(d)}\log(p(X_i=1,X_{j(i)}=0,C=1)) \\
& + \sum_{d=1}^{N}\sum_{i=s+1}^{n}x_i^{(d)}x_{j(i)}^{(d)}(1-c^{(d)})\log(p(X_i=1,X_{j(i)}=1,C=0)) \\
& + \sum_{d=1}^{N}\sum_{i=s+1}^{n}x_i^{(d)}x_{j(i)}^{(d)}c^{(d)}\log(p(X_i=1,X_{j(i)}=1,C=1))
\end{aligned}
$$

Matching the latter expression with the terms in Equation 5.12 yields:

$$\boldsymbol{\mathcal{U}} = u(C, \boldsymbol{X}) = \left(\sum_{d=1}^{N} c^{(d)}, \sum_{d=1}^{N} c^{(d)} x_1^{(d)}, \ldots, \sum_{d=1}^{N} c^{(d)} x_n^{(d)}, \right.$$
$$\left. \sum_{d=1}^{N} c^{(d)} x_{s+1}^{(d)} x_{j(s+1)}^{(d)}, \ldots, \sum_{d=1}^{N} c^{(d)} x_n^{(d)} x_{j(n)}^{(d)} \right)$$

$$\boldsymbol{\mathcal{V}} = v(\boldsymbol{X}) = \left(\sum_{d=1}^{N} x_1^{(d)}, \ldots, \sum_{d=1}^{N} x_n^{(d)}, \sum_{d=1}^{N} x_{s+1}^{(d)} x_{j(s+1)}^{(d)}, \ldots, \sum_{d=1}^{N} x_n^{(d)} x_{j(n)}^{(d)} \right)$$

and then, the sufficient statistics are given by:

$$\boldsymbol{\mathcal{U}} = (N_0, N_{01}, \ldots, N_{0i}, \ldots, N_{0n}, N_{0(s+1)j(s+1)}, \ldots, N_{0nj(n)})$$
$$\boldsymbol{\mathcal{V}} = (N_1, \ldots, N_i, \ldots, N_n, N_{(s+1)j(s+1)}, \ldots, N_{nj(n)})$$

with:

$$N_0 = \#\{C = 1\} = \sum_{d=1}^{N} c^{(d)}$$
$$N_{0i} = \#\{C = 1, X_i = 1\} = \sum_{d=1}^{N} c^{d} x_i^{(d)} \quad i = 1, \ldots, n$$
$$N_{0ij(i)} = \#\{C = 1, X_i = 1, X_{j(i)} = 1\} = \sum_{d=1}^{N} c^{d} x_i^{(d)} x_{j(i)}^{(d)} \quad i = s+1, \ldots, n$$
$$N_i = \#\{X_i = 1\} = \sum_{d=1}^{N} x_i^{(d)} \quad i = 1, \ldots, n$$
$$N_{ij(i)} = \#\{X_i = 1, X_{j(i)} = 1\} = \sum_{d=1}^{N} x_i^{(d)} x_{j(i)}^{(d)} \quad i = s+1, \ldots, n$$

$E[\boldsymbol{\mathcal{U}}|\boldsymbol{x}]$, at the $t+1$-th iteration of the algorithm, is calculated as:

$$E_{\boldsymbol{\theta}^{(t)}}[N_0|\boldsymbol{x}] = \sum_{d=1}^{N} p(C=1|\boldsymbol{X}^{(d)}=\boldsymbol{x}^{(d)},\boldsymbol{\theta}^{(t)}) \tag{5.40}$$

$$E_{\boldsymbol{\theta}^{(t)}}[N_{0i}|\boldsymbol{x}] = \sum_{d=1}^{N} p(C=1|\boldsymbol{X}^{(d)}=\boldsymbol{x}^{(d)},\boldsymbol{\theta}^{(t)})x_i^{(d)} \quad i=1,2,\ldots,n$$

$$E_{\boldsymbol{\theta}^{(t)}}[N_{0ij(i)}|\boldsymbol{x}] = \sum_{d=1}^{N} p(C=1|\boldsymbol{X}^{(d)}=\boldsymbol{x}^{(d)},\boldsymbol{\theta}^{(t)})x_i^{(d)}x_{j(i)}^{(d)} \quad i=s+1,\ldots,n$$

As well as in the TM algorithm for naive Bayes with dichotomic variables, in order to compute the expected sufficient statistics (Equation 5.40) it is needed to calculate the conditional probability of the class variable given the predictive variables. Bearing in mind the conditional model defined in Equation 5.38, the following conditional probabilities are needed:

$$p(C=1) = \frac{N_0^{(t)}}{N}$$

$$p(X_i=1|C=1) = \frac{N_{0i}^{(t)}}{N_0^{(t)}}$$

$$p(X_i=1|C=0) = \frac{N_i - N_{0i}^{(t)}}{N - N_0^{(t)}}$$

$$p(X_i=1|X_{j(i)}=1,C=1) = \frac{N_{0ij(i)}^{(t)}}{N_{0j(i)}^{(t)}}$$

$$p(X_i=1|X_{j(i)}=1,C=0) = \frac{N_{ij(i)} - N_{0ij(i)}^{(t)}}{N_{j(i)} - N_{0j(i)}^{(t)}}$$

$$p(X_i=1|X_{j(i)}=0,C=1) = \frac{N_{0i}^{(t)} - N_{0ij(i)}^{(t)}}{N_0^{(t)} - N_{0j(i)}^{(t)}}$$

$$p(X_i=1|X_{j(i)}=0,C=0) = \frac{N_i - N_{ij(i)}^{(t)} - (N_{0i}^{(t)} - N_{0ij(i)}^{(t)})}{N - N_{j(i)} - (N_0^{(t)} - N_{0j(i)}^{(t)})}$$

Finally, the sufficient statistics and the parameters of the FAN model are calculated, at iteration $t+1$, in the same way as described in Section 5.2.2.1.

5.2.2.3 The TM Algorithm for BAN Models with Dichotomic Variables

In this section, we present the adaptation of the TM algorithm for BAN models. In the case of BAN models, a predictive variable can have, at most, $n-$

1 parents besides the class variable. Hence, in order to simplify the notation, we represent this set of variables as $\boldsymbol{Pa}_i = \{Z_{i1}, \ldots, Z_{ij}, \ldots, Z_{ih_i}\}$, where Z_{ij} is the j-th parent of X_i, with $i = 1, \ldots, n$ and $h_i = |\boldsymbol{Pa}_i|$ the number of parents of X_i.

The conditional BAN model can be written as follows:

$$p(c|\boldsymbol{x}) = \frac{p(c)\prod_{i=1}^{n} p(x_i|c, z_{i1}, \ldots, z_{ih_i})}{p(\boldsymbol{x})} \tag{5.41}$$

and the joint model is given by:

$$p(c, \boldsymbol{x}) = p(c)\prod_{i=1}^{n} \frac{p(c, x_i, z_{i1}, \ldots, z_{ih_i})}{p(c, z_{i1}, \ldots, z_{ih_i})} \tag{5.42}$$

After a transformation in the log-likelihood function similar to the one exposed in Sections 5.2.2.1 and 5.2.2.2, it is possible to obtain the expression of $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$ functions:

$$\boldsymbol{\mathcal{U}} = \Bigg(\sum_{d=1}^{N} c^{(d)}, \sum_{d=1}^{N} c^{(d)} z_{ij_1}^{(d)} \ldots, z_{ij_{n_j}}^{(d)}, \sum_{d=1}^{N} c^{(d)} x_i^{(d)} z_{ij_1}^{(d)} \ldots, z_{ij_{n_j}}^{(d)} | i = 1, \ldots, n$$
$$, \{j_1, \ldots, j_{n_j}\} \in \mathcal{P}\{1, 2, \ldots, h_i\} \setminus \emptyset\Bigg)$$

$$\boldsymbol{\mathcal{V}} = \Bigg(\sum_{d=1}^{N} z_{ij_1}^{(d)} \ldots, z_{ij_{n_j}}^{(d)}, \sum_{d=1}^{N} x_i^{(d)} z_{ij_1}^{(d)} \ldots, z_{ij_{n_j}}^{(d)} | i = 1, \ldots, n$$
$$, \{j_1, \ldots, j_{n_j}\} \in \mathcal{P}\{1, 2, \ldots, h_i\} \setminus \emptyset\Bigg)$$

where $\mathcal{P}\{1, 2, \ldots, h_i\} \setminus \emptyset$ is the power set of $\{1, 2, \ldots, h_i\}$ excluding the empty set. That is, the set of all subsets in $\{1, 2, \ldots, h_i\}$ excluding the empty set. Moreover, the sufficient statistics related to $\boldsymbol{\mathcal{U}}$ are:

$$N_0 = \#\{C = 1\} = \sum_{d=1}^{N} c^{(d)}$$

$$N_{0z_{ij_1}\ldots z_{ij_{n_j}}} = \#\{C = 1, Z_{ij_1} = 1 \ldots, Z_{ij_{n_j}} = 1\} = \sum_{d=1}^{N} c^{(d)} z_{ij_1}^{(d)} \cdots z_{ij_{n_j}}^{(d)}$$

$$N_{0x_i z_{ij_1}\ldots z_{ij_{n_j}}} = \#\{C = 1, X_i = 1, Z_{ij_1} = 1 \ldots, Z_{ij_{n_j}} = 1\}$$
$$= \sum_{d=1}^{N} c^{(d)} x_i^{(d)} z_{ij_1}^{(d)} \cdots z_{ij_{n_j}}^{(d)}$$

where $\{j_1, \ldots, j_{n_j}\} \in \mathcal{P}\{1, 2, \ldots, h_i\} \setminus \emptyset$ with $i = 1, 2, \ldots, n$.
Additionally, the sufficient statistics related to $\boldsymbol{\mathcal{V}}$ are:

$$N_{z_{ij_1}\ldots z_{ij_{n_j}}} = \#\{Z_{ij_1} = 1 \ldots, Z_{ij_{n_j}} = 1\} = \sum_{d=1}^{N} z_{ij_1}^{(d)} \cdots z_{ij_{n_j}}^{(d)}$$

$$N_{x_i z_{ij_1}\ldots z_{ij_{n_j}}} = \#\{X_i = 1, Z_{ij_1} = 1 \ldots, Z_{ij_{n_j}} = 1\} = \sum_{d=1}^{N} x_i^{(d)} z_{ij_1}^{(d)} \cdots z_{ij_{n_j}}^{(d)}$$

where also $\{j_1, \ldots, j_{n_j}\} \in \mathcal{P}\{1, 2, \ldots, h_i\} \setminus \emptyset$ with $i = 1, 2, \ldots, n$.

The calculation of $E[\boldsymbol{\mathcal{U}}|\boldsymbol{x}]$ is similar to the one for naive Bayes and FAN models:

$$E_{\boldsymbol{\theta}^{(t)}}[N_0|\boldsymbol{x}] = \sum_{d=1}^{N} p(C = 1|\boldsymbol{X} = \boldsymbol{x}^{(d)}, \boldsymbol{\theta}^{(t)})$$

$$E_{\boldsymbol{\theta}^{(t)}}[N_{0z_{ij_1}\ldots z_{ij_{n_j}}}|\boldsymbol{x}] = \sum_{d=1}^{N} p(C = 1|\boldsymbol{X} = \boldsymbol{x}^{(d)}, \boldsymbol{\theta}^{(t)}) z_{ij_1}^{(d)} \ldots z_{ij_{n_j}}^{(d)}$$

$$E_{\boldsymbol{\theta}^{(t)}}[N_{0x_i z_{ij_1}\ldots z_{ij_{n_j}}}|\boldsymbol{x}] = \sum_{d=1}^{N} p(C = 1|\boldsymbol{X} = \boldsymbol{x}^{(d)}, \boldsymbol{\theta}^{(t)}) x_i^{(d)} z_{ij_1}^{(d)} \ldots z_{ij_{n_j}}^{(d)}$$

In order to obtain the expected sufficient statistics, the calculation of several conditional probabilities are needed. However, although the computation of $p(C)$ is simple,

$$p(C = 1) = \frac{N_0^{(t)}}{N} = \frac{\#\{C = 1\}}{N} \tag{5.43}$$

the computation of the conditional probabilities $p(x_i|c, z_{i1}, \ldots, z_{ih_i})$ becomes more complex due to the number of parents that a predictive variable can have.

In order to illustrate the calculation of $p(x_i|c, z_{i1}, \ldots, z_{ih_i})$, we show how the first values are obtained. Hence, the calculation of the remainder conditional probabilities can be performed in a similar manner. However, it is important to perform the calculations in a proper order, thus we are able to re-use previous calculations when computing new ones. For instance, the conditional probabilities for the variable X_i where all its parents but one take the value 1 can be computed as follows:

$$p(X_i = 1|C = 1, Z_{i1} = 1, \ldots, Z_{ih_i} = 1)$$

$$= \frac{N^{(t)}_{0x_i z_{i1} \ldots z_{ih_i}}}{N_{0 z_{i1} \ldots z_{ih_i}}}$$

$$= \frac{\#\{X_i{=}1, C{=}1, Z_{i1}{=}1, \ldots, Z_{ih_i}{=}1\}}{\#\{C{=}1, Z_{i1}{=}1, \ldots, Z_{ih_i}{=}1\}}$$

$$p(X_i = 1|C = 1, Z_{i1} = 0, Z_{i2} = 1, \ldots, Z_{ih_i} = 1)$$

$$= \frac{N^{(t)}_{0x_i z_{i2} \ldots z_{ih_i}} - N^{(t)}_{0x_i z_{i1} \ldots z_{ih_i}}}{N_{0, z_{i2} \ldots z_{ih_i}} - N_{0 z_{i1} \ldots z_{ih_i}}}$$

$$= \frac{\#\{X_i{=}1, C{=}1, Z_{i1}{=}0, Z_{i2}{=}1 \ldots, Z_{ih_i}{=}1\}}{\#\{C{=}1, Z_{i1}{=}0, Z_{i2}{=}1, \ldots, Z_{ih_i}{=}1\}}$$

$$\vdots$$

$$p(X_i = 1|C = 1, Z_{i1} = 1, \ldots, Z_{ih_i - 1} = 1, Z_{ih_i} = 0)$$

$$= \frac{N^{(t)}_{0x_i z_{i1} \ldots z_{ih_i - 1}} - N^{(t)}_{0x_i z_{i1} \ldots z_{ih_i}}}{N_{0 z_{i1} \ldots z_{ih_i - 1}} - N_{0 z_{i1} \ldots z_{ih_i}}}$$

$$= \frac{\#\{X_i{=}1, C{=}1, Z_{i1}{=}1, \ldots, Z_{ih_i - 1}{=}1, Z_{ih_i}{=}0\}}{\#\{Z_{i1}{=}1, \ldots, Z_{ih_i - 1}{=}1, Z_{ih_i}{=}0\}}$$

Now, $p(X_i = 1|C = 1, Z_{i1} = 0, Z_{i2} = 0, Z_{i3} = 1, \ldots, Z_{ih_i} = 1)$ can be computed by using previous calculations. Thus, the numerator to calculate $p(X_i = 1|C = 1, Z_{i1} = 0, Z_{i2} = 0, Z_{i3} = 1, \ldots, Z_{ih_i} = 1)$ can be obtained as follows:

$$\begin{aligned}
&\#\{X_i = 1, C = 1, Z_{i1} = 0, Z_{i2} = 0, Z_{i3} = 1, \ldots, Z_{ih_i - 1} = 1, Z_{ih_i} = 1\} \\
&\quad = \#\{X_i = 1, C = 1, Z_{i3} = 1, \ldots, Z_{ih_i} = 1\} \\
&\qquad -\#\{X_i = 1, C = 1, Z_{i1} = 1, Z_{i2} = 0, Z_{i3} = 1, \ldots, Z_{ih_i} = 1\} \\
&\qquad -\#\{X_i = 1, C = 1, Z_{i1} = 0, Z_{i2} = 1, Z_{i3} = 1, \ldots, Z_{ih_i} = 1\} \\
&\qquad -\#\{X_i = 1, C = 1, Z_{i1} = 1, Z_{i2} = 1, Z_{i3} = 1, \ldots, Z_{ih_i 0} = 1\} \quad (5.44)
\end{aligned}$$

where the first term in Equation 5.44 is the sufficient statistic $N^{(t)}_{0x_i z_{i3} \ldots z_{ih_i}}$. On the other hand, the second and third terms in Equation 5.44 have been previously calculated. Finally, the fourth term in Equation 5.44 is the sufficient statistic $N^{(t)}_{0x_i z_{i1} \ldots z_{ih_i}}$. Similarly, the denominator in $p(X_i = 1|C = 1, Z_{i1} = 0, Z_{i2} = 0, Z_{i3} = 1, \ldots, Z_{ih_i} = 1)$ can be obtained from sufficient statistics and previously calculated values:

$$\begin{aligned}
&\#\{C=1, Z_{i1}=0, Z_{i2}=0, Z_{i3}=1, \ldots, Z_{ih_i-1}=1, Z_{ih_i}=1\} \\
&\quad = \#\{C=1, Z_{i3}=1, \ldots, Z_{ih_i}=1\} \\
&\quad\quad -\#\{Z_{i1}=1, Z_{i2}=0, Z_{i3}=1, \ldots, Z_{ih_i}=1\} \\
&\quad\quad -\#\{Z_{i1}=0, Z_{i2}=1, Z_{i3}=1, \ldots, Z_{ih_i}=1\} \\
&\quad\quad -\#\{Z_{i1}=1, Z_{i2}=1, Z_{i3}=1, \ldots, Z_{ih_i}=1\}
\end{aligned} \tag{5.45}$$

In general, let's assume, without loss of generality that, at iteration t,

$$p(X_i=1|C=1, Z_{i1}=0, Z_{i2}=0, \ldots, Z_{is}=0, Z_{is+1}=1, Z_{is+2}=1, \ldots, Z_{ih_i}=1) \tag{5.46}$$

is calculated. The numerator can be obtained by the following equation:

$$\#\{X_i=1, C=1, Z_{i1}=0, Z_{i2}=0, \ldots, Z_{is}=0, Z_{is+1}=1, Z_{is+2}=1, \ldots, Z_{ih_i}=1\} =$$

$$\#\{X_i=1, C=1, Z_{is+1}=1, Z_{is+2}=1, \ldots, Z_{ih_i}=1\} - \sum_{z_{i1}, z_{i2}, \ldots, z_{is}=0,1 | \sum_{j=1}^{s} z_{ij} \neq 0}$$
$$\#\{X_i=1, C=1, Z_{i1}=z_{i1}, Z_{i2}=z_{i2}, \ldots, Z_{is}=z_{is}, Z_{is+1}=1, Z_{is+2}=1, \ldots, Z_{ih_i}=1\}$$

where the summation includes all the possible combinations for the values $z_{i1}, z_{i2}, \ldots, z_{is}$ except for the case where $z_{i1} = z_{i2} = \ldots = z_{is} = 0$. Again, the quantities needed in the calculations have been previously computed and they correspond with sufficient statistics.

Analogously, the denominator in Equation 5.46 can be obtained as follows:

$$\#\{C=1, Z_{i1}=0, Z_{i2}=0, \ldots, Z_{is}=0, Z_{is+1}=1, Z_{is+2}=1, \ldots, Z_{ih_i}=1\}$$

$$= \#\{C=1, Z_{is+1}=1, Z_{is+2}=1, \ldots, Z_{ih_i}=1\} - \sum_{z_{i1}, z_{i2}, \ldots, z_{is}=0,1 | \sum_{j=1}^{s} z_{ij} \neq 0}$$
$$\#\{C=1, Z_{i1}=z_{i1}, Z_{i2}=z_{i2}, \ldots, Z_{is}=z_{is}, Z_{is+1}=1, Z_{is+2}=1, \ldots, Z_{ih_i}=1\}$$

Note that, all the calculations shown is this section are related to $C = 1$. However, the calculations where $C = 0$ can be obtained is a similar manner.

Once all the calculations introduced in this section have been obtained, the t-th iteration of the TM algorithm for a BAN model with dichotomic variables is identical to the one described in Section 5.2.2.1.

5.2.2.4 The TM Algorithm for Naive Bayes Models with Multinomial Variables

In the case of multinomial variables, each variable can take multiple states. Therefore, in order to simplify the notation we assume $C \in \{0, \ldots, r_0\}$ and

$X_i \in \{0, \ldots, r_i\}$ with $r_0 + 1$ and $r_i + 1$ as the number of possible states for variables C and X_i, respectively. Note, that in other parts of this dissertation we refer to the class variable with a C subindex (for example r_C) but here we simplify this notation by using 0 as the index of the class variable.

The general TM algorithm for probability distributions in the exponential family requires the expression of the joint log-likelihood via Equation 5.12. This can be achieved by writing the naive Bayes joint model as follows:

$$p(c, \boldsymbol{x}) = \frac{1}{(p(c))^{n-1}} \prod_{i=1}^{n} p(x_i, c) \tag{5.47}$$

In order to identify the sufficient statistics for the TM algorithm, we can rewrite the joint model as:

$$p(c, \boldsymbol{x}) = \left[\prod_{j=0}^{r_0} (p(C=j))^{w_j \prod_{l=0}^{j-1}(c-l) \prod_{l=j+1}^{r_0}(l-c)} \right]^{-(n-1)} \cdot$$

$$\prod_{i=1}^{n} \prod_{j=0}^{r_0} \prod_{k=0}^{r_i} (p(C=j, X_i=k))^{w_{jk}^i \prod_{l=0}^{j-1}(c-l) \prod_{l=j+1}^{r_0}(l-c) \prod_{l=0}^{k-1}(x_i-l) \prod_{l=k+1}^{r_i}(l-x_i)}$$

where w_j and w_{jk}^i are the following constants:

$$w_j = \frac{1}{\prod_{l=0}^{j-1}(j-l) \prod_{l=j+1}^{r_0}(l-j)}$$

$$w_{jk}^i = \frac{1}{\prod_{l=0}^{j-1}(j-l) \prod_{l=j+1}^{r_0}(l-j) \prod_{l=0}^{k-1}(k-l) \prod_{l=k+1}^{r_i}(l-k)}$$

Note that the values of w_j and w_{jk}^i have no influence on the selection of the sufficient statistics for the TM algorithm.

If we have a dataset with N samples, the log-likelihood can be written using the previous equation in the following manner:

$$\begin{aligned} l(\boldsymbol{\theta}) = \sum_{d=1}^{N} \Bigg\{ & -(n-1) \sum_{j=0}^{r_0} \left[w_j \prod_{l=0}^{j-1} (c^{(d)} - l) \prod_{l=j+1}^{r_0} (l - c^{(d)}) \log(p(C=j)) \right] + \\ & \sum_{i=1}^{n} \sum_{j=0}^{r_0} \sum_{k=0}^{r_i} \Big[w_{jk}^i \prod_{l=0}^{j-1} (c^{(d)} - l) \prod_{l=j+1}^{r_0} (l - c^{(d)}) \\ & \prod_{l=0}^{k-1} (c_i^{(d)} - l) \prod_{l=k+1}^{r_i} (l - c_i^{(d)}) \log(p(C=j, X_i=k)) \Big] \Bigg\} \end{aligned} \tag{5.48}$$

where $c^{(d)}$ and $x_i^{(d)}$ are the values of variables C and X_i in the d-th sample of the dataset, respectively.

A few transformations in Equation 5.48 can match its terms with those from Equation 5.12. We thus obtain the sufficient statistics $\boldsymbol{\mathcal{U}} = (\boldsymbol{\mathcal{U}}_1, \boldsymbol{\mathcal{U}}_2)$ and $\boldsymbol{\mathcal{V}}$:

$$\begin{aligned}
\boldsymbol{\mathcal{U}}_1 &= (M_0^s | s = 1, \ldots, r_0) \\
\boldsymbol{\mathcal{U}}_2 &= (M_{0x_i}^{sm} | s = 1, \ldots, r_0 \ , \quad m = 1, \ldots, r_i \ , \quad i = 1, \ldots, n) \\
\boldsymbol{\mathcal{V}} &= (M_{x_i}^m | m = 1, \ldots, r_i \ , \quad i = 1, \ldots, n)
\end{aligned}$$

where M_0^s, $M_{0x_i}^{sm}$ and $M_{x_i}^m$ terms from the former equation are defined as:

$$\begin{aligned}
M_0^s &= \sum_{d=1}^{N} (c^{(d)})^s \\
M_{0x_i}^{sm} &= \sum_{d=1}^{N} (c^{(d)})^s (x_i^{(d)})^m \\
M_{x_i}^{(d)} &= \sum_{r=1}^{N} (x_i^{(d)})^m
\end{aligned} \tag{5.49}$$

It was shown in Section 5.2.1.2 that, at each iteration, the calculation of $E_{\boldsymbol{\theta}}\{\boldsymbol{\mathcal{U}} | \boldsymbol{x}\}$ is needed in order to update the sufficient statistics $\boldsymbol{\mathcal{U}}$. This requires the following calculations:

$$\begin{aligned}
E_{\boldsymbol{\theta}^{(t)}}[M_0^s | \boldsymbol{x}] &= \sum_{d=1}^{N} \sum_{c=0}^{r_0} p(C = c | \boldsymbol{X} = \boldsymbol{x}^{(d)}, \boldsymbol{\theta}^{(t)}) c^s \\
E_{\boldsymbol{\theta}^{(t)}}[M_{0x_i}^{sm} | \boldsymbol{x}] &= \sum_{d=1}^{N} \sum_{c=0}^{r_0} p(C = c | \boldsymbol{X} = \boldsymbol{x}^{(d)}, \boldsymbol{\theta}^{(t)}) c^s (x_i^{(d)})^m
\end{aligned} \tag{5.50}$$

where $s = 1, \ldots, r_0$; $m = 1, \ldots, r_i$ and $i = 1, \ldots, n$.

Since we assume that the structure of the model is a naive Bayes, we need to obtain $p(C = c)$ and $p(X_i = l | C = c)$ to calculate $p(C = c | \boldsymbol{X} = \boldsymbol{x}^{(d)})$, where $c = 0, \ldots, r_0$; $i = 1, \ldots, n$ and $l = 0, \ldots, r_i$. In order to obtain these probabilities, let us define a new set of sufficient statistics $\boldsymbol{\mathcal{N}} = (N_0^c, N_i^l, N_{0i}^{cl} | c = 0, \ldots, r_0; \ \ i = 1, \ldots, n; \ \ l = 0, \ldots, r_i)$. On the one hand, N_0^c counts the number of cases in which $C = c$, and N_i^l the number of cases in which $X_i = l$. On the other hand, N_{0i}^{cl} denotes the number of cases where both $C = c$ and $X_i = l$ happen.

The sufficient statistics in $\boldsymbol{\mathcal{N}}$ are related to the sufficient statistics from the TM algorithm, $(\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{V}})$. In the special case where all the variables are

dichotomic, both sets of sufficient statistics, $\boldsymbol{\mathcal{N}}$ and $\boldsymbol{\mathcal{U}}$, are the same. However, when the variables are multinomial, this relationship is given by linear systems of equations which can be obtained by means of Equation 5.49. Therefore, using these systems of equations we are able to obtain the values of $\boldsymbol{\mathcal{N}}$ from $\boldsymbol{\mathcal{U}}$ and vice versa.

As an example, we show how one of the linear systems of equations can be obtained from Equation 5.49. M_0^s, with $s = 1, \ldots, r_0$, are sufficient statistics from $\boldsymbol{\mathcal{U}}$ and, as it was written in Equation 5.49, $M_0^s = \sum_{d=1}^{N} (C^{(d)})^s$. Since $\sum_{d=1}^{N} (C^{(d)}) = 0 \cdot N_0^0 + \ldots + r_0 \cdot N_0^{r_0}$, the system of equation that relates both $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{N}}$ for the variable C can be written in matrix form as follows:

$$\underbrace{\begin{pmatrix} 1 & 2 & \ldots & r_0 \\ 1 & 2^2 & \ldots & r_0^2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 2^{r_0} & \ldots & r_0^{r_0} \end{pmatrix}}_{\boldsymbol{COEFFS^*}} \underbrace{\begin{pmatrix} N_0^1 \\ N_0^2 \\ \vdots \\ N_0^{r_0} \end{pmatrix}}_{\boldsymbol{\mathcal{N}^*}} = \underbrace{\begin{pmatrix} M_0^1 \\ M_0^2 \\ \vdots \\ M_0^{r_0} \end{pmatrix}}_{\boldsymbol{\mathcal{U}^*}} \tag{5.51}$$

Once we have obtained the values of the statistics in $\boldsymbol{\mathcal{N}}$, we are able to calculate $p(C = c)$ and $p(X_i = l|C = c)$ by:

$$p(C = c) = \frac{N_0^c}{N}$$
$$p(X_i = l|C = c) = \frac{N_{0i}^{cl}}{N_0^c}$$

and then calculate the value of $E_{\boldsymbol{\theta}}\{\boldsymbol{\mathcal{U}}|\boldsymbol{x}\}$. Finally, we are able to iterate the algorithm and thus obtain the new values for the statistics in $\boldsymbol{\mathcal{U}}$ (see Equation 5.18). These $p(C = c)$ and $p(X_i = l|C = c)$ are also the parameters $\boldsymbol{\theta}$ of the naive Bayes classifier that we are learning. Hence, we have to calculate $\boldsymbol{\mathcal{N}}$ in order to obtain $\boldsymbol{\theta}$. A general algorithm for the discriminative learning of Bayesian classifiers with multinomial variables is given in Figure 5.2.

The process of maximizing the conditional log-likelihood with the TM algorithm looks computationally hard because we have to solve several linear systems of equations at each iteration. However, from one iteration of the algorithm to another one, in the systems of equations, only the values in $\boldsymbol{\mathcal{U}^*}$ change (see Equation 5.51). Therefore, we can obtain the LU transformation of $\boldsymbol{COEFFS^*}$, which is constant throughout the algorithm. Thus, the solution of the system of equations at each iteration is quite simple. Moreover, the LU transformation is also the same for every problem with the same number of variables and the same number of states per variable. Hence, it may be feasible to calculate these transformations and store the solutions for future use.

Obtain $\boldsymbol{n}^{(0)}$ from the dataset

Calculate $\boldsymbol{u}^{(0)}$ from $\boldsymbol{n}^{(0)}$

while stopping criterion is not met **do**

  Calculate $E_{\boldsymbol{\theta}^{(t)}}\{\boldsymbol{\mathcal{U}}|\boldsymbol{x}\}$

  Update $\boldsymbol{u}$:

    $\boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t)} + \boldsymbol{u}^{(0)} - E_{\boldsymbol{\theta}^{(t)}}\{\boldsymbol{\mathcal{U}}|\boldsymbol{x}\}$

  Calculate $\boldsymbol{n}^{(t+1)}$ from $\boldsymbol{u}^{(t+1)}$

  Calculate $\boldsymbol{\theta}^{(t+1)}$ from $\boldsymbol{n}^{(t+1)}$

  if illegal $\boldsymbol{\theta}^{(t+1)}$ or conditional log-likelihood decreases **then**

    Find the best legal $\boldsymbol{\theta}^{(t+1)}$ via linear search

  end if

end while

Fig. 5.2. General pseudo-code for the discriminative learning of Bayesian classifiers with multinomial variables. Note that $\boldsymbol{n}^{(t)}$ and $\boldsymbol{u}^{(t)}$ are the values of the statistics in $\boldsymbol{\mathcal{N}}$ and $\boldsymbol{\mathcal{U}}$ at iteration t, respectively

5.2.2.5 The TM algorithm for FAN models with multinomial variables

In this section we introduce the adaptation of the TM algorithm for FAN models where the variables are assumed to be multinomial. The development of the TM algorithm for FAN models assumes that the structure of the model is already known. Therefore, before performing the discriminative learning of a FAN model, we need to set its structure.

The adaptation of the TM algorithm introduced here is similar to the adaptation for a naive Bayes model shown in the previous section. As well as in the TM algorithm for FAN models with dichotomic variables, we assume, without lost of generality, that the first s variables have only one parent (the class variable) and the parent set for the reminder $n-s$ variables is composed of the class and another predictive variable. Therefore, the conditional and joint FAN model expressions are identical to the ones introduced in Section 5.2.2.2 (see Equations 5.38 and 5.39).

If we develop the $l(\boldsymbol{\theta})$ function for a FAN model with multinomial variables in a similar way to Equation 5.12, we can identify the following set of sufficient statistics $\boldsymbol{\mathcal{U}} = (\boldsymbol{\mathcal{U}}_1, \boldsymbol{\mathcal{U}}_2, \boldsymbol{\mathcal{U}}_3)$ and $\boldsymbol{\mathcal{V}} = (\boldsymbol{\mathcal{V}}_1, \boldsymbol{\mathcal{V}}_2)$:

$$\boldsymbol{\mathcal{U}_1} = (M_0^w | w = 1, 2, \ldots, r_0)$$
$$\boldsymbol{\mathcal{U}_2} = (M_{0x_i}^{wm} | w = 1, 2, \ldots, r_0;\ m = 1, 2, \ldots, r_i;\ i = 1, \ldots, n)$$
$$\boldsymbol{\mathcal{U}_3} = (M_{0x_i x_{j(i)}}^{wmz} | w = 1, \ldots, r_0;\ m = 1, 2, \ldots, r_i;\ z = 1, 2, \ldots, r_{j(i)};\ i = s+1, \ldots, n)$$
$$\boldsymbol{\mathcal{V}_1} = (M_{x_i}^{m} | m = 1, \ldots, r_i)$$
$$\boldsymbol{\mathcal{V}_2} = (M_{x_i x_{j(i)}}^{mz} | m = 0, 1, \ldots, r_i;\ z = 0, 1, \ldots, r_{j(i)};\ i = s+1 \ \ldots, n)$$

with M_c^w, $M_{cx_i}^{wm}$, $M_{cx_i x_{j(i)}}^{wmz}$, $M_{x_i}^m$ and $M_{x_i x_{j(i)}}^{mz}$ defined as follows:

$$M_c^w = \sum_{d=1}^{N} (c^{(d)})^w$$
$$M_{cx_i}^{wm} = \sum_{d=1}^{N} (c^{(d)})^w (x_i^{(d)})^m$$
$$M_{cx_i x_{j(i)}}^{wmz} = \sum_{d=1}^{N} (c^{(d)})^w (x_i^{(d)})^m (x_{j(i)}^{(d)})^z$$
$$M_{x_i}^{m} = \sum_{d=1}^{N} (x_i^{(d)})^m$$
$$M_{x_i x_{j(i)}}^{mz} = \sum_{d=1}^{N} (x_i^{(d)})^m (x_{j(i)}^{(d)})^z$$

The calculation of the expected sufficient statistics are obtained by:

$$\boldsymbol{\mathcal{V}_1} = (M_{x_i}^{m} | m = 1, \ldots, r_i, i = 1, \ldots, n)$$
$$\boldsymbol{\mathcal{V}_2} = (M_{x_i x_{j(i)}}^{mz} | m = 0, 1, \ldots, r_i\ ,\ \ z = 0, 1, \ldots, r_{j(i)}\ ,\ \ i = s+1 \ \ldots, n)$$

In order to calculate the expected sufficient statistics, we need to compute $p(c)$, $p(x_i|c)$ and $p(x_i|c, x_{j(i)})$. $p(c)$ and $p(x_i|c)$ can be computed as in Section 5.2.2.4 but for the computation of $p(x_i|c, x_{j(i)})$, we must define, for each variable X_i, a new set of sufficient statistics denoted as $N_{0ij(i)}^{clq}$. Then, $N_{0ij(i)}^{clq}$ represents the number of samples where variable C takes the value c (with $c = 0, \ldots, r_0$), X_i takes the value l (with $l = 0, \ldots, r_i$), and $X_{j(i)}$ takes the value q (with $q = 0, \ldots, r_{j(i)}$). Thus, $p(x_i|c, x_{j(i)})$ can be calculated as:

$$p(X_i = l | C = c, X_{j(i)} = q) = \frac{\#\{C = c, X_i = l, X_{j(i)} = q\}}{\#\{C = c, X_{j(i)} = q\}} \qquad (5.52)$$

where the numerator is the sufficient statistic $N_{0ij(i)}^{clq}$ and the denominator has been previously obtained in the calculation of $p(x_{j(i)}|c)$. Additionally, the

sufficient statistics $N_{0ij(i)}^{clq}$ can be obtained from $\boldsymbol{\mathcal{U}}$ by solving the following system of equations:

$$\begin{aligned}
\sum_{c=1}^{r_0}\sum_{l=1}^{r_i}\sum_{q=1}^{r_{j(i)}} clqN_{0ij(i)}^{clq} &= M_{0x_ix_{j(i)}}^{111}\\
\sum_{c=1}^{r_0}\sum_{l=1}^{r_i}\sum_{q=1}^{r_{j(i)}} c^2lqN_{0ij(i)}^{clq} &= M_{0x_ix_{j(i)}}^{211}\\
&\vdots = \vdots\\
\sum_{c=1}^{r_0}\sum_{l=1}^{r_i}\sum_{q=1}^{r_{j(i)}} c^{r_0}lqN_{0ij(i)}^{clq} &= M_{0x_ix_{j(i)}}^{r_011}\\
\sum_{c=1}^{r_0}\sum_{l=1}^{r_i}\sum_{q=1}^{r_{j(i)}} cl^2qN_{0ij(i)}^{clq} &= M_{0x_ix_{j(i)}}^{121}\\
&\vdots = \vdots\\
\sum_{c=1}^{r_0}\sum_{l=1}^{r_i}\sum_{q=1}^{r_{j(i)}} c^{r_0}l^{r_i}qN_{0ij(i)}^{clq} &= M_{0x_ix_{j(i)}}^{r_0r_i1}\\
\sum_{c=1}^{r_0}\sum_{l=1}^{r_i}\sum_{q=1}^{r_{j(i)}} clq^2N_{0ij(i)}^{clq} &= M_{0x_ix_{j(i)}}^{112}\\
&\vdots = \vdots\\
\sum_{c=1}^{r_0}\sum_{l=1}^{r_i}\sum_{q=1}^{r_{j(i)}} c^{r_0}l^{r_i}q^{r_{j(i)}}N_{0ij(i)}^{clq} &= M_{0x_ix_{j(i)}}^{r_0r_ir_{j(i)}}
\end{aligned}$$

with $i = s+1, \ldots, n$.

Note that in the previous system of equations the sufficient statistics where $c = 0$ and/or $q = 0$ is not calculated but they can be obtained in the same manner as the sufficient statistics for a naive Bayes model with multinomial variables (see Section 5.2.2.4). Finally, the rest of the TM algorithm for FAN models with multinomial variables performs as described in the general pseudo-code from Figure 5.2.

5.2.3 Experimental Results for the Discriminative Learning of Parameters

In previous sections, the theoretical development of the TM algorithm for naive Bayes, FAN and BAN models with dichotomic variables and for naive Bayes and FAN models with multinomial variables has been introduced. In

this section we aim to provide an empirical evaluation which attempts to illustrate the performance of the TM algorithm applied to Bayesian classification models. We firstly evaluate the TM algorithm in datasets with dichotomic variables and then in datasets with multinomial variables. The datasets with dichotomic variables are simpler and therefore they allow a more exhaustive evaluation by means of a *leaving-one-out* cross-validation. By contrast, the evaluation of the TM algorithm in datasets with multinomial variables is computationally more expensive. Hence, we use a five-fold cross-validation in order to evaluate the accuracy of the classifiers. This five-fold cross-validation scheme has been previously used in the literature to evaluate the generative (Friedman *et al.*, 1997) and discriminative (Greiner *et al.*, 2005) learning of Bayesian network classifiers.

The TM algorithm iteratively maximizes the conditional log-likelihood. In the experiments, the algorithm stops when the difference between the conditional log-likelihood value in two consecutive steps is less than 0.001. On the other hand, as pointed out in Section 5.2.1.2, the TM calculations may lead the parameters of the model to illegal values. These situations are solved by applying a local search with $\lambda = 0.01$ (Equation 5.19).

In the case of naive Bayes models, the structure does not depend on the data, that is, a naive Bayes model may differ from another one only in the predictive variables included in the model. However, the structure of both TAN and BAN models is learned from the dataset with different criteria. The TAN structure is learned by using the algorithm proposed in Friedman *et al.* (1997), which takes into account the conditional mutual information of two variables given the class. On the other hand, the structure of the BAN classifier, in principle, can be learned by using any structural learning algorithm for Bayesian networks but we have used the K2 algorithm introduced in Chapter 2. In the experiments, the order over the predictive variables for the K2 algorithm is taken at random and the maximum number of parents has been set to four. Note that the K2 algorithm learns the structure of the model only with the predictive variables. Afterwards, the class variable is set as a parent of all of them. Hence, although BAN is a classification model, the learning of the BAN structure is not oriented to classification problems.

We are aware that the K2 is a greedy algorithm and that the order over the predictive variables influences the final result to a large extent (Larrañaga *et al.*, 1996). Therefore, the proposed learning method may not find the best model structure for the classifier. However, the goal of these experiments is not to find the best classifier but to compare the performance of a classifier whose parameters are learned with the TM algorithm with the one with maximum likelihood parameters.

5.2.3.1 Experimental Results with Dichotomic Variables

For the experiments with dichotomic variables, we have used four different datasets obtained from MLC++ repository at `http://www.sgi.com/tech/mlc/`.

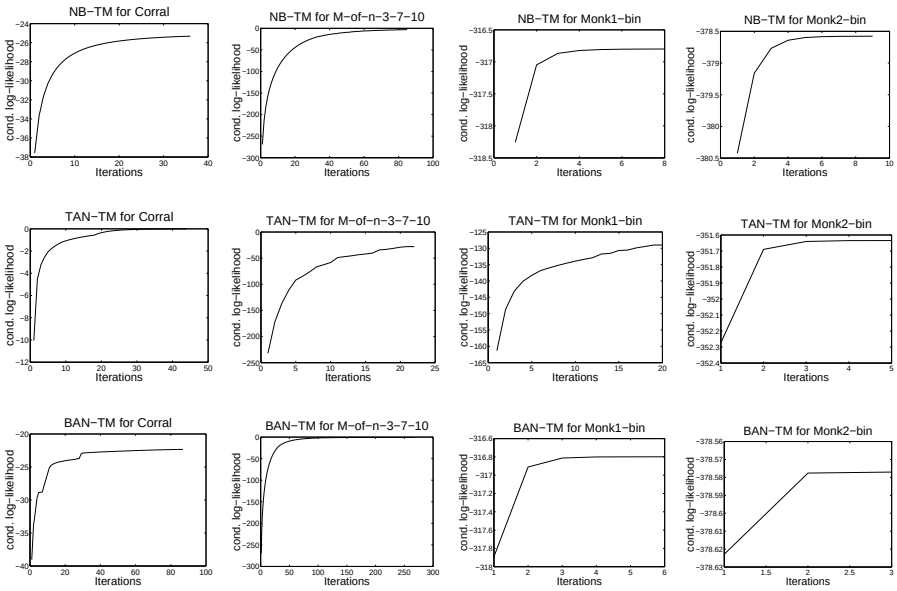


Fig. 5.3. Conditional log-likelihood values for the experiments with naive Bayes, TAN and BAN models

These datasets are `Corral`, `M-of-n-3-7-10`, `Monk1-bin` and `Monk2-bin`. Some of them, such as `Monk1-bin` and `Monk2-bin`, are adaptations from the original UCI datasets (Blake and Merz, 1998).

As it was said before, a leaving-one-out cross-validation is used to measure the estimated accuracy of both classifiers learned with the TM algorithm and classifiers with maximum likelihood parameters. On the other hand, the structure learning for TAN and BAN models depends on some decisions which are taken at random: the selection of the root for the tree in the case of TAN, and the order over the predictive variables in the case of BAN. Therefore, the resulting structure may change among different learnings. Hence, in order to compare the classifiers, the structures of these models are learned only once for each dataset and using all the instances. Then, these structures are maintained for the generative and discriminative learning of the parameters.

Figure 5.3 shows how the conditional log-likelihood value for models learned using the whole dataset converges throughout the iterations of the algorithm for each data set. In the plots we see that the stopping criterion used in the experiments seems appropriate because the conditional log-likelihood does not seem to improve much more even if we ran more iterations of the algorithm. On the other hand, although, as expected, the evolution of the conditional log-likelihood value throughout each experiment is monotonously increasing in all of them, the TAN model for `Corral`, `M-of-n-3-7-10` and `Monk1-bin`, and the BAN model for `Corral` present some irregular increas-

	NB	*NB–TM*	*TAN*	*TAN–TM*	*BAN*	*BAN–TM*
`Corral`	-33.25	-25.10	-10.05	-0.005	-39.03	-22.33
`M-of-n-3-7-10`	-269.32	-2.88	-232.03	-27.93	-269.31	-0.16
`Monk1-bin`	-318.26	-316.79	-161.32	-128.97	-317.79	-316.80
`Monk2-bin`	-393.77	-378.84	-352.27	-351.63	-378.62	-378.58

Table 5.1. Conditional log-likelihood values for the experiments with naive Bayes, TAN and BAN models.

	NB	*NB–TM*	*TAN*	*TAN–TM*	*BAN*	*BAN–TM*
`Corral`	84.37±3.21	90.62±**2.58**	100.00±0.00	100.00±0.0	56.25±0.50	56.25±0.50
`M-of-n-3-7-10`	84.21±1.00	100.00±**0.00**	89.20±0.31	100.00±**0.00**	77.95±0.41	77.95±0.41
`Monk1-bin`	60.79±0.49	63.67±**0.48**	82.01±**0.38**	78.78±0.41	49.64±0.50	51.44±**0.50**
`Monk2-bin`	61.57±2.34	67.13±**2.26**	63.06±0.48	63.89±0.48	65.72±0.47	65.72±0.47

Table 5.2. Estimated accuracy obtained in the experiments with naive Bayes and TAN models.

ing which may be caused by the local search method (Equation 5.19). Additionally, for each final model, the conditional log-likelihood value for the discriminative and generative models is also shown in Table 5.1.

In Table 5.2 the accuracy for naive Bayes, TAN, and BAN classifiers learned from `Corral`, `M-of-n-3-7-10`, `Monk1-bin`, and `Monk2-bin` datasets are shown. Additionally, for each model we compare the results of the discriminative and generative learning by using a Mann-Whitney test. We write the results in Table 5.2 in bold if the test is surpassed with a p-value < 0.05 and in a gray color box if the test is surpassed with a p-value < 0.01. The TM algorithm improves the estimated accuracy for naive Bayes in all the datasets. This may be because naive Bayes is a very simple classifier and it does not perfectly model the data. Therefore, discriminative learning by the TM algorithm is more efficient than generative learning. Thus, it improves the accuracy of the classifier. When the structure of the model is more complex, such as the case of TAN models, the difference between the estimated accuracy for the generative and the discriminative model decreases. Even the generative model beats the discriminative one in the case of `Monk1-bin` dataset. On the other hand, both generative and discriminative BAN classifiers (Table 5.2) obtain the same accuracy results in all data sets except `Monk1-bin`, where the discriminative learning is significantly better. Moreover, the BAN classifiers obtain less expected accuracy than TAN and naive Bayes models. This may be caused by the learning process used for the BAN structure. The reader may remember that the order among the predictive variables is taken at random and that the class variable is not taken into account in the learning

process of the structure. This may also explain the bad performance of the BAN classifiers with respect to naive Bayes and TAN.

5.2.3.2 Experimental Results with Multinomial Variables

In this section we present some experiments which attempt to illustrate the performance of the TM algorithm used to learn Bayesian classification models such as naive Bayes and TAN from datasets with multinomial variables.

We have evaluated the TM algorithm for the discriminative learning of Bayesian classifiers using sixteen datasets obtained from the UCI repository (Blake and Merz, 1998). Moreover, we use the `Corral` and `M-of-n-3-7-10` datasets, which were developed by Kohavi and John (1997) to evaluate methods for subset selection, and the `Tips` dataset (Inza *et al.*, 2001c). `Tips` is a medical dataset to identify the subgroup of patients surviving within the first six months after the *transjugular intrahepatic portosystemic shunt* (TIPS) placement, a non-surgical method to avoid portal hypertension.

Note that `Corral` and `M-of-n-3-7-10` datasets only contain dichotomic variables and they were used in the experiments with dichotomic variables. However, we decided to include these datasets in the current experiments to evaluate how the cross-validation method used in the experiments affects the obtained results.

The discriminative learning of Bayesian network classifiers by means of the TM algorithm does not deal with missing data or continuous variables. Therefore, a pre-processing step was needed before using the datasets. On the one hand, every data sample which contained missing data was removed. On the other hand, variables with continuous values were discretized using Fayyad and Irani's (Fayyad and Irani, 1993) discretization method. The accuracy of the classifiers is measured by five-fold cross-validation and it is based on the percentage of successful predictions. The same pre-processing and validation methodology has been used before in the literature for the generative (Friedman *et al.*, 1997) and discriminative (Greiner *et al.*, 2005) learning of Bayesian network classifiers using all the datasets that have been used in this section, except for `Tips`.

Similarly to the experiments with dichotomic variables, the algorithm stops when the difference between the conditional log-likelihood value in two consecutive steps is less than 0.001. On the other hand, as pointed out in Section 5.2.1.2, the TM calculations may lead the parameters of the model to illegal values. These situations are solved by applying a linear search where we look for λ in interval $(0, 1)$ with a 0.01 increment (Equation 5.19).

Table 5.3 shows the estimated accuracy for the naive Bayes (NB) and TAN classifiers learned using both generative and discriminative approaches. The generative approach that we use in the experiments is the maximum likelihood estimation of the parameters. In contrast, the discriminative learning is carried out using the TM algorithm presented in this chapter. In order to

	NB	NB–TM	TAN	TAN-TM
Australian	85.65±2.61	88.41±2.67	86.08±2.88	88.98±3.53
Breast	97.37±1.64	98.98±0.74	97.37±1.64	95.46±1.41
Chess	87.77±0.91	95.15±0.41	92.40±1.73	96.81±0.49
Cleve	83.14±4.89	**87.53±4.72**	82.77±1.61	87.85±3.24
Corral	86.77±9.27	90.61±6.27	100.00±0.00	99.20±1.60
Crx	86.68±4.70	88.52±1.59	86.06±1.33	**89.59±1.56**
Flare	92.12±2.16	95.12±1.21	95.78±2.79	96.72±1.23
German	75.40±3.50	**78.90±4.00**	72.80±2.22	84.00±0.89
Glass	74.31±7.32	76.18±6.92	72.90±2.74	81.75±3.88
Heart	83.33±6.73	86.67±4.44	72.90±2.74	81.75±3.87
Hepatitis	85.00±10.15	93.75±5.56	87.50±6.84	100.00±0.00
Iris	94.67±3.40	95.33±3.40	93.33±2.11	96.00±2.49
Lymphography	83.77±4.97	91.22±3.49	79.08±2.28	98.98±1.65
M-of-n-3-7-10	86.63±2.53	100.00±0.00	90.86±1.79	100.00±0.00
Pima	77.96±1.31	79.95±1.47	79.17±3.72	79.82±3.72
Soybean-large	96.26±1.64	97.51±1.42	98.58±0.71	99.29±0.66
Tips	88.78±4.65	100.00±0.00	89.87±6.20	100.00±0.00
Vehicle	61.94±1.58	78.61±1.51	71.63±4.19	83.46±3.72
Vote	89.88±2.45	98.39±1.17	93.56±1.55	99.08±0.86

Table 5.3. Estimated accuracy obtained in the experiments with naive Bayes and TAN models.

compare the estimated accuracy for both discriminative and generative models we perform a Mann-Whitney test, whose results are also shown in Table 5.3. We write in bold those values that surpass the test with a p-value < 0.1 and in a gray color box those values that surpass the test with a p-value < 0.05. The TM algorithm improves the estimated accuracy for naive Bayes and TAN models in all the datasets but two. In fact, in `Breast` and `Corral` datasets the generative learning of TAN models obtains a higher estimated accuracy, although only for `Breast` the difference between TAN and TAN-TM is statistically significant. This may be caused by the fact that the TAN model is complex enough to model the relationship among the variables which are present in the `Breast` dataset, and therefore, the generative learning can outperform the discriminative learning. Nevertheless, even when the estimated accuracy is usually higher in discriminative models, the difference with respect to generative models is not always significant. In most of the cases the fact that the improvement obtained by the discriminative method is not significant with a p-value < 0.05 is caused by the high standard deviation. A cause of this high standard deviation may be the small number of folds used in the cross-validation process. Indeed we can compare the results for `Corral` and `M-of-n-3-7-10` with those obtained from a leaving-one-out cross-validation (Section 5.2.3.1). In the case of `M-of-n-3-7-10` the results with five-fold cross-validation and leaving-one-out cross-validation are identical for discriminative

	NB	NB–TM	TAN	TAN-TM
Australian	-291.71	-190.38	-195.97	-195.97
Breast	-136.74	-22.71	-22.13	-10.41
Chess	-917.76	-478.22	-591.46	-307.04
Cleve	-124.65	-93.24	-96.36	-82.27
Corral	-37.58	-25.17	-10.19	-3.28
Crx	-251.71	-177.90	-173.71	-173.71
Flare	-287.91	-216.18	-152.76	-137.43
German	-488.91	-456.81	-409.05	-378.12
Glass	-141.86	-141.86	-117.60	-116.96
Heart	-112.91	-86.11	-88.21	-73.66
Hepatitis	-23.29	-9.27	-9.12	-2.61
Iris	-21.52	-13.85	-17.90	-16.57
Lymphography	-46.94	-28.06	-25.90	-13.17
M-of-n-3-7-10	-269.32	-3.75	-232.03	-44.74
Pima	-361.36	-340.55	-333.24	-331.22
Soybean-large	-108.15	-43.27	-22.14	-22.14
Tips	-45.66	-0.12	-2.77	-0.03
Vehicle	-1487.18	-355.30	-360.27	-297.16
Vote	-257.63	-13.66	-49.55	-13.88

Table 5.4. Conditional log-likelihood values for the experiments with naive Bayes and TAN models.

learning. By contrast, the leaving-one-out cross-validation always leads, for Corral and generative models learned from M-of-n-3-7-10, to an important decrease in the standard deviation while the estimated accuracy does not change so much. Nevertheless, although a more expensive validation method such as a ten-fold cross-validation may be computationally feasible for these experiments, we decided to maintain the cross-validation schema used in the literature (Friedman *et al.*, 1997; Greiner *et al.*, 2005; Santafé *et al.*, 2005b) in order to have a point of reference for the results obtained in our experiments. Nonetheless, it is difficult to compare the results obtained in these experiments with those from Greiner *et al.* (2005) because the data needed to perform a statistical test is not provided in the paper but the TM learning seems to obtain slightly better results than Greiner et al.'s method in most of these datasets.

On the other hand, the aim of the TM algorithm is to maximize the conditional log-likelihood. In Table 5.4, the improvement of the conditional log-likelihood score for the discriminative model with respect to the generative one is shown. As described in Section 5.2.1.1, the TM algorithm begins with the same parameters obtained by the generative model (that is the maximum likelihood parameters) and, following an iterative process, it modifies these parameters to maximize the conditional log-likelihood. Note that the TM algorithm is able to obtain a model with higher value for the conditional log-likelihood score in all datasets except for the TAN model learned from Australian, Crx and Soybean-large. This is because, in these three cases,

the parameters that maximize the joint log-likelihood also represent a maximum for the conditional log-likelihood score. This maximum is not necessarily a global maximum but may be a local one. The case of `Crx` dataset is a clear example of bias in the cross-validation process because even when generative and discriminative TAN are the same models, the difference between the estimated accuracies is significant with a p-value < 0.1. This can be possible because the conditional log-likelihood value reported in Table 5.4 is obtained from a classifier which has been learned using the whole dataset. On the other hand, for the cross-validation process, whose results are shown in Table 5.3, the classifiers are learned using only part of the dataset. Therefore, for each fold, generative and discriminative classifiers may differ one from each other.

5.3 Discriminative Learning of Structures for Bayesian Network Classifiers

The discriminative learning of structures for Bayesian network classifiers is a topic that has been recently addressed by a few publications. Probably, the first approach to structural discriminative learning for Bayesian network classifiers is presented in Grossman and Domingos (2004) where the structure of the classifier is learned by a wrapper greedy approach and using a discriminative score. This score is the conditional BIC (cBIC) metric (see Equation 5.53), which is a modification of the BIC score by using the conditional log-likelihood instead of the log-likelihood.

$$cBIC = \sum_{d=1}^{N} \log p(c^{(d)}|\boldsymbol{x}^{(d)}, \mathcal{B}) - \frac{1}{2} \log N dim(S) \tag{5.53}$$

where $\mathcal{B} = (S, \boldsymbol{\theta})$ and $dim(S)$ is the number of parameters needed to totally specify the joint model $p(c, \boldsymbol{x})$ given the factorization encoded by the Bayesian network structure S.

However, the parameters of the model are determined by their maximum likelihood estimation, which is a generative parameter learning approach. Guo and Greiner (2005) compare generative scores to learn the structure of Bayesian network classifiers such as BIC and marginal likelihood with other discriminative scores such as cBIC, classification error on training data and $(bias)^2 + variance$ decomposition of the expected mean-square error of the classifier (Ripley, 1996). Additionally, the parameter learning of the models is not only performed by using the maximum likelihood approximation, but also by using the discriminative method proposed in Greiner *et al.* (2005).

Narasimhan and Bilmes (2005) propose an algorithm to minimize the difference between two submodular functions using a variational framework based on the concave-convex procedure (Yuille and Rangarajan, 2002). This algorithm is used to maximize the explaining away residual (EAR)

$$EAR(X_i, X_j, C) = I(X_i, X_j|C) - I(X_i, X_j) = -I(X_i, X_j, C) \quad (5.54)$$

which approximates the conditional log-likelihood score, in order to discriminatively learn tree structures. Additionally, they propose the use of a greedy algorithm to optimize the EAR score when learning TAN or general Bayesian networks for classification. Moreover, the discriminative parameter learning is performed in a similar manner to Greiner *et al.* (2005).

In following sections two algorithms for the discriminative learning of Bayesian network classifiers are presented. On the one hand, we present a modification of Friedman et al.'s algorithm that allows to maximize the EAR score to learn TAN structures (Pernkopf and Bilmes, 2005; Perez *et al.*, 2006). On the other hand, we present the structural TM algorithm (Santafé *et al.*, 2005a, 2006a) which extends the TM algorithm to learn the structure of a Bayesian network classifier.

5.3.1 Learning Conditional TAN Models

The construct-discriminative-TAN algorithm (Pernkopf and Bilmes, 2005; Perez *et al.*, 2006) is a recently proposed algorithm to learn the structure of TAN models from a discriminative point of view. This algorithm is a modification of Friedman et al's algorithm (Friedman *et al.*, 1997) in order to maximize the EAR score (McGill, 1954; Watanabe, 1960)(see Equation 5.54). For TAN structures, maximizing the EAR measure is equivalent to maximizing the conditional log-likelihood. This can be seen in the following theorem borrowed from (Perez *et al.*, 2007):

Theorem 4. *Given a dataset D with N instances and given an estimator of $p(.)$ for the data (maximum likelihood estimation), the maximization of the conditional likelihood is equivalent to maximizing $I(C, \boldsymbol{X})$.*

Proof. Using the empirical distribution $\hat{p}(c, \boldsymbol{x}) = \frac{1}{N}$ iff $(c, \boldsymbol{x}) \in D$ and 0 otherwise, we can rewrite the CLL (see Equation 3.8) in terms of entropy (Jebara, 2003; Cover and Thomas, 2006):

$$\begin{aligned} CLL &= N E_{\hat{p}(c,\boldsymbol{x})}\big(\log p(c|\boldsymbol{x})\big) = -N(H_{\hat{p}(c,\boldsymbol{x})}(C|\boldsymbol{X})) \\ &\propto -H_{\hat{p}(c,\boldsymbol{x})}(C) + I_{\hat{p}(c,\boldsymbol{x})}(C, \boldsymbol{X}) \end{aligned} \quad (5.55)$$

The term $H_{\hat{p}(c,\boldsymbol{x})}(C)$ is constant for each specific problem and then, the conditional log-likelihood is proportional to $I_{\hat{p}(c,\boldsymbol{x})}(C, \boldsymbol{X})$. Therefore, the maximization of $I_{\hat{p}(c,\boldsymbol{x})}(C, \boldsymbol{X})$ implies maximizing the conditional log-likelihood. Using the chain rule for the mutual information (Cover and Thomas, 2006), it can be written as follows:

Compute $-I(X_i, X_j, C)$ for $i < j$ and $i, j = 1, \ldots, n$

Let G be a complete undirected graph where $-I(X_i, X_j, C)$ is the weight of edge $X_i - X_j$

Use Kruskal algorithm to obtain the maximum spanning tree from G

Select a random root to set the direction of the edges

Add the class variable as parent of each predictive variable

Fig. 5.4. General pseudo-code for construct-discriminative-TAN algorithm.

$$I(\boldsymbol{X}, C) = \sum_{i=1}^{n} I(X_i, C|\boldsymbol{Pa_i}) \tag{5.56}$$

Taking into consideration the structural restrictions for TAN models, Equation 5.56 can be re-written as:

$$I(\boldsymbol{X}, C) = \sum_{i=1}^{n} I(X_i, C|\boldsymbol{Pa_i}) = \sum_{i=1}^{n} I(X_i, C) - \sum_{i=1}^{n} I(X_i, X_{j(i)}, C) \tag{5.57}$$

where $X_{j(i)}$ is the predictive variable parent of X_i and $I(X_i, X_{j(i)}, C)$ equals zero when X_i is the root of the tree, that is, X_i has no predictive variables as parents. Note that, the first term in Equation 5.57 is constant. Therefore, to maximizing the conditional log-likelihood is equivalent to minimize the term $\sum_{i=1}^{n} I(X_i, X_{j(i)}, C)$. □

Therefore, as the conditional log-likelihood can be maximized by maximizing the joint mutual information of the predictive and the class variables, a simple filter greedy algorithm can be obtained by adapting Friedman et al.'s construct-TAN to maximize $I(\boldsymbol{X}, C)$ instead of $I(\boldsymbol{X}|C)$. This construct-discriminative-TAN algorithm is given in Figure 5.4.

Note that, the construct-discriminative-TAN algorithm (Figure 5.4) obtains a TAN structure that maximizes the conditional log-likelihood. However, the algorithm uses the mutual information $I(X_i, X_j, C)$ which differs from $I(X_i, X_j)$ and $I(X_i, X_j|C)$ in the fact that it can take either positive or negative values. Therefore, as we force a tree structure in the algorithm, some of the included links may contribute to decrease the conditional log-likelihood score. That is, in contrast to log-likelihood score where the more complex the model structure is, the higher the log-likelihood score becomes, conditional log-likelihood may decrease as the complexity of the model increases. This can be easily avoided by forbidding arcs with $-I(X_i, X_j, C) < 0$ (Pernkopf and Bilmes, 2005; Perez *et al.*, 2006). Nevertheless, this process may yield a FAN structure instead of a TAN structure. Hence, we prefer to maintain the TAN restrictions and learn tree-like structures for a fair comparison between TAN and cTAN models.

5.3.2 Structural TM Algorithm

In this section we extend the algorithm TM to learn the structure of a Bayesian network classifier. The structural TM (Santafé *et al.*, 2005a, 2006a) is a greedy wrapper algorithm which starts with an empty structure. At each step of the algorithm, there is a set of candidate arcs which can be added to the network. This set is formed by all the arcs which are not in the network structure and whose addition to the network structure does not violate the structural restrictions of the model, for instance the addition of the arc does not create a cycle in the network. Thus, the parameters for each proposed structure are learned by using the TM algorithm and the model is scored by using cBIC metric (Equation 5.53). The structural TM algorithm is described in Figure 5.5. Note that, the proposed approximation differs from the one introduced in Grossman and Domingos (2004) in two main concepts. On the one hand, we use the TM algorithm to learn the parameter of each candidate solution, that is, we perform a discriminative learning of both structure and parameters of the classifier. By contrast, although in Grossman and Domingos (2004) the structure is learned by using a discriminative approach, the parameters are learned from a generative approach. On the other hand, as the proposed TM algorithm learns the parameters for the conditional model $p(C|\boldsymbol{X})$, we use the dimension of the conditional model for the penalty term in cBIC score (Equation 5.53). Other discriminative approaches (Greiner *et al.*, 2005; Roos *et al.*, 2005) do not allow to identify the parameters of the conditional model. Therefore, Grossman and Domingos (2004) use the dimension of the joint model for the penalty of the cBIC score. By contrast, the formulation of the TM algorithm allows to identify the parameters of the conditional model (see Equation 5.12) and then, we use the number of parameters in the conditional model as penalty term for the cBIC score.

5.3.3 Experimental Results for Discriminative Learning of Structures

In this section we present some experiments to evaluate the performance of the proposed discriminative structure learning approaches.

On the one hand, we use the algorithm described in Section 5.3.1 in order to learn conditional TAN structures using UCI datasets.

On the other hand, we show how the structural TM algorithm introduced in Section 5.3.2 performs on some UCI dataset. Note that for these experiments we only use `Cleve`, `Iris`, `German`, `Lymphography`, `Hepatitis`, `Vote` and `Heart` datasets from UCI as well as `Corral` dataset from Kohavi and John (1997). Due to the wrapper nature of the structural TM algorithm, the computational cost of learning models with many variables is very high. Therefore, we select the datasets taking into account the moderate number of variables that they present. Additionally, as the TM algorithm implementation is able to learn the parameters of Bayesian network classifiers up to TAN models,

```
Let S be an empty structure with n variables and no arcs
Obtain the list L of candidate arcs that can be added to S
while not stop do
    for each arc A in L do
        Add A to S
        Learn the parameters using the TM algorithm
        Evaluate the addition of A to S by cBIC
        Remove A from S
    end for
    Let A_max be the arc from L which addition to S maximizes cBIC
    if the inclusion of A_max in S increases cBIC from previous iteration do
        Add A to S
    else
        stop
    end if
end while
```

Fig. 5.5. General pseudo-code for structural TM.

the structural search is also limited by TAN complexity. Another constraint imposed by the implementation of the TM algorithm is that it can not deal with either continuous variables or missing values. Therefore, following the same methodology as in the empirical evaluation of the TM algorithm to learn the parameters for Bayesian network classifiers, every data sample containing missing data is removed and continuous variables are discretized using Fayyad and Irani's discretization method (Fayyad and Irani, 1993).

The structural TM algorithm learns the parameters of each candidate model in the searching process by using the TM algorithm. Similarly, the parameter for the TAN structure obtained with the construct-discriminative-TAN algorithm are learned with the TM algorithm. Therefore, the parameters of the TM algorithm must be set. In the experiments, the stopping criterion for the TM algorithm is met when the difference between the conditional log-likelihood value in two consecutive steps is less than 0.001. Additionally, the use of the TM algorithm may involve the use of a linear search method to correct illegal parameter sets or the decrease of the conditional log-likelihood score (Section 5.2.1.2). This linear search looks for λ in interval $(0, 1)$ with a 0.01 increment (see Equation 5.19). The models are evaluated by means of a

five-fold cross-validation process. This validation process has also been used in this dissertation to evaluate the TM algorithm. Table 5.5 shows the accuracy values obtained by the models whose structure has been learned with the structural TM and construct-discriminative-TAN algorithms and the parameters with the TM algorithm (structural-TM and cTAN-TM respectively). Moreover, the accuracy values for other models such as NB-TM, TAN-TM are also shown for comparison. We can see from Table 5.5 that, in general, the TAN-TM model obtain better results than the rest of the models. Even when the discriminative parameter learning of TAN models (TAN-TM) is better than the generative parameter learning (TAN) (see Table 5.3) the discriminative learning of structure and parameters (cTAN-TM) usually obtain worse estimated accuracy values. This may be caused by the fact that the discriminative-construct-TAN algorithm includes arcs in the model to form a TAN model even when some of these arcs can penalize the conditional log-likelihood value. As said in Section 5.3.1, we include these arcs in order to obtain a TAN instead of a FAN structure. With respect to the structural TM algorithm, it obtains similar results to the cTAN-TM models. It obtains a higher estimated accuracy for `Iris`, `Corral`, `Vote` and `Heart` and worse results than the cTAN-TM model for the rest of datasets. However, the models obtained by the structural TM algorithm are simpler than the other models because they do not include all the predictive variables (Figure 5.6 shows the structures learned by the structural TM algorithm using all the data samples in the dataset). Therefore, the models learned by the structural TM are sometimes even simpler than NB-TM models[1]. Then, they may contribute to an easier interpretation of the model and the problem. See Table 5.6 for a comparison between the number of variables in each dataset and the number of variables selected by structural TM models.

	Structural-TM	cTAN-TM	NB-TM	TAN-TM
`Cleve`	82.42±4.89	83.44±5.18	87.53±4.72	87.85±3.24
`Iris`	95.33±3.40	94.00±4.35	95.33±3.40	96.00±2.49
`German`	73.30±3.75	74.80±2.97	78.90±4.00	84.00±0.89
`Corral`	98.46±3.08	91.42±8.47	90.61±6.27	99.20±1.60
`Lymphography`	71.61±10.92	85.15±5.15	91.22±3.49	98.98±1.65
`Hepatitis`	87.50±7.90	85.75±7.13	93.75±5.56	100.00±0.00
`Vote`	95.63±2.45	92.41±1.74	98.39±1.17	99.08±0.86
`Heart`	85.56±2.46	84.94±7.00	86.67±4.44	81.75±3.87

Table 5.5. Accuracy values for the experiments with structural-TM, cTAN-TM, NB-TM and TAN-TM models.

[1] The obtained models for `Cleve`, `Iris`, `German`, `Lymphography`, `Hepatitis`, `Vote` and `Heart` can be seen as selective naive Bayes models

	Variables Dataset	*Variables Structural-TM Model*
`Cleve`	13 + Class	5 + Class
`Iris`	4 + Class	1 + Class
`German`	20 + Class	4 + Class
`Corral`	6 + Class	5 + Class
`Lymphography`	18 + Class	4 + Class
`Hepatitis`	19 + Class	2 + Class
`Vote`	16 + Class	3 + Class
`Heart`	13 + Class	4 + Class

Table 5.6. Number of predictive variables for each dataset and number of predictive variables selected by the structural TM algorithm.

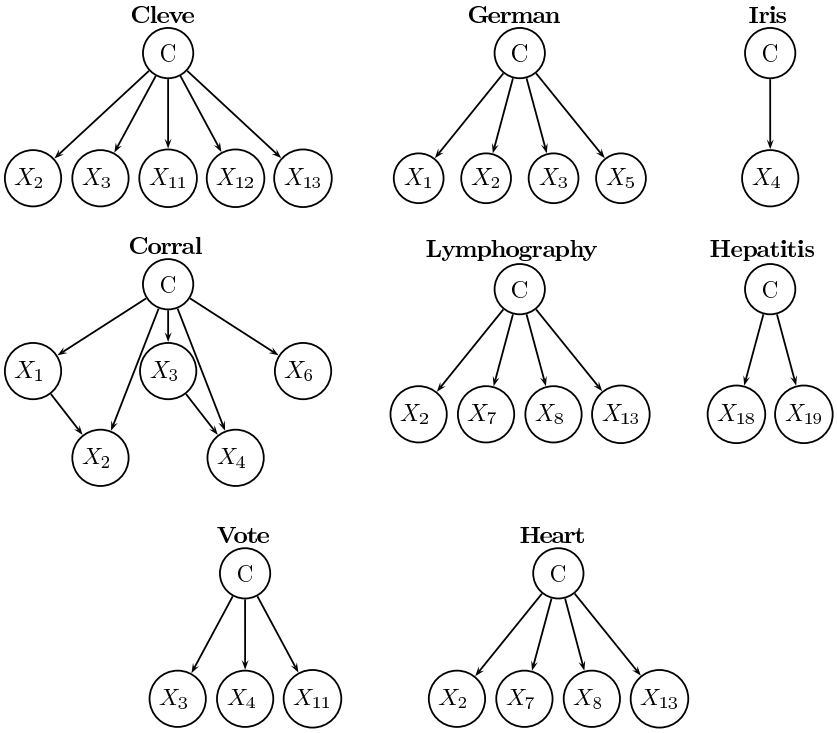


Fig. 5.6. Graphical structures of the classifiers learned with structural TM algorithm using all the samples in the dataset.

The reader may note that in these experiments we have not performed any statistical test to compare the results obtained by the different algorithms. This is because, in every experiment, both structural TM and cTAN-TM models obtain worse results than NB-TM or TAN-TM models. Hence, even when these differences may not always be statistically significant, algorithms

such as structural TM are computationally much more expensive than obtaining a naive Bayes structure or than generatively learning of the structure by means of the construct-TAN algorithm (Friedman *et al.*, 1997). Therefore, these learning methods for the structures may be preferred.

5.4 Generative vs. Discriminative Learning

In this section we present several experiments with both synthetic and real dataset from UCI (Blake and Merz, 1998). These experiments attempt to illustrate some of the ideas about generative and discriminative learning introduced in Section 5.1.

5.4.1 Synthetic Datasets

As was pointed out before, it is thought that if the modeling assumptions (or restrictions) set when learning a Bayesian network classifier from data are incorrect, a discriminative approach should be preferred because the bias is smaller. In this section we attempt to illustrate this behavior by learning classification models from datasets which have been sampled from random models with different structural complexity (NB, TAN and $p(c, \boldsymbol{x})$ models). The number of predictive variables varies in $\{8,12,16\}$, each predictive variable takes up to three states and the number of classes vary in $\{2,3\}$. For each configuration, we generate 10 random models and each one of these models is simulated 50 times to obtain 50 different datasets with 40, 80, 120, 200, 300, 400 and 500 samples. This process yields 500 different datasets for each model configuration and dataset size. A five-fold cross-validation schema is used to estimate the classification rate of NB, TAN and cTAN classifiers on each dataset with both generative (maximum likelihood) and discriminative (TM algorithm) approaches to learn the parameters. The models used in these experiments may seem quite simple to the reader since the number of variables is not very high. The computational resources required to deal with the joint probability model prevent us from using more variables. Nevertheless, we think that these models are able to illustrate the performance of both generative and discriminative learning approaches.

Figure 5.7 shows how naive Bayes, TAN and cTAN models learned with both discriminative and generative parameter learning approaches perform, on average, in datasets sampled from random naive Bayes models. We can see in the plots that a naive Bayes model learned with a generative parameter learning approach performs better than the rest of the models. This is because a naive Bayes is enough to capture all the relationships between variables. By contrast, TAN and cTAN models are overestimating the relationships between variables, they create artificial relationships that are not really represented in the dataset and that may lead them to obtain a worse classification rate than the naive Bayes model. Additionally, discriminative learning of the parameters

for TAN models seems to perform slightly better than generative learning because the relationships between variables assumed by the TAN model are not true in this case. However, we can not appreciate relevant differences between discriminative and generative structural learning.

Similarly, Figure 5.8 shows the results for datasets sampled from random TAN models. In this case, naive Bayes can not capture the relations between predictive variables. Hence, naive Bayes models obtain a worse classification rate than TAN and cTAN models. The performance of TAN and cTAN models is, again, very similar when the parameters are learned by a generative method, but a discriminative learning of the parameters for cTAN models (that is, discriminative learning of both structure and parameters) obtains, in this experiment, a worse classification rate. This behavior of the cTAN-TM models is surprising because the obtained results are much worse than those obtained with cTAN models with generative parameter learning. This behavior may be caused by the fact that the conditional log-likelihood for cTAN models, when the structure is learned from datasets sampled from TAN models, may present many local maxima. Thus, the TM algorithm can be trapped in one of these local maxima and therefore the algorithm is not able to properly maximize the conditional log-likelihood in this particular case which may lead the model to obtain poor accuracy results.

In general, we can see that, as was expected, the generative learning performs better (or very similar) than the discriminative approach when the restrictions in the model that we are learning agree with the model used to generate the dataset.

On the other hand, we would also like to test how naive Bayes, TAN and cTAN classifiers behave when the datasets are generated by using more complex models such as joint probability distributions, $p(c, \boldsymbol{x})$ (see Figure 5.9). In this experiment, the structural restrictions of the models that we are learning from the datasets (naive Bayes, TAN and cTAN) do not agree with the model that generates the data. Therefore, discriminative learning performs better, in terms of classification rate, than generative learning, at least for the parameter learning. However, the discriminative learning of TAN structures (cTAN models) performs very similar to the generative learning of the structure (TAN models).

5.4.2 UCI Repository

In this section, we develop a simple experiment that illustrates the use of the log-likelihood (LL) and conditional log-likelihood (CLL) to guide the learning process of the parameters for a naive Bayes, TAN and cTAN models in problems obtained from UCI repository.

Once the structure of the model is learned with the corresponding method, we obtain ten thousand different parameter sets at random and evaluate the log-likelihood, conditional log-likelihood and the classification rate of each classifier in the datasets. Then, we plot log-likelihood vs. classification rate

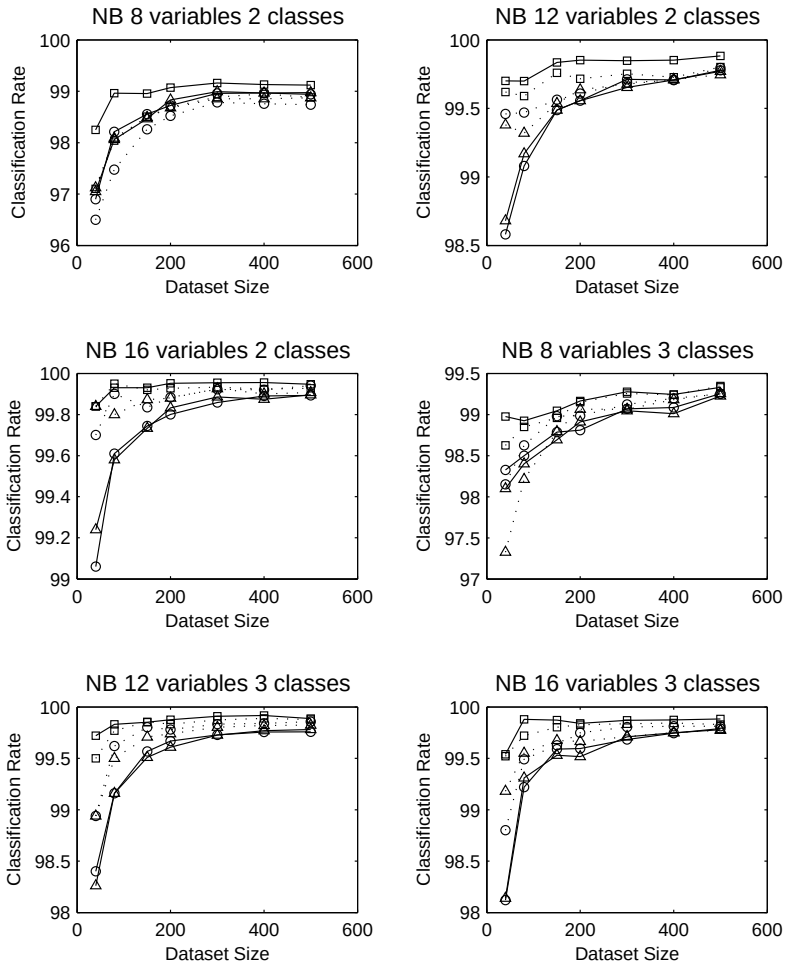


Fig. 5.7. Experiments with datasets sampled from random NB models. Solid lines represent generative learning while dotted lines discriminative learning. □ denotes NB models; △ TAN models and ∘ cTAN models. Each point in the plot represents the classification rate on average over 500 different datasets.

and conditional log-likelihood vs. classification rate to evaluate the tendency of both log-likelihood and conditional log-likelihood scores with respect to the classification rate (Figures 5.10, 5.11, 5.12, 5.13, 5.14, 5.15 and 5.16). The results shown in this section represent the general behavior that we have observed among different UCI datasets and they give the reader some intuition about the relation between generative learning, discriminative learning and the classification rate. Additionally, Table 5.7 shows the Spearman's correlation coefficient between the log-likelihood and the classification rate for each model (NB gen., TAN gen. and cTAN gen.) and between the conditional log-

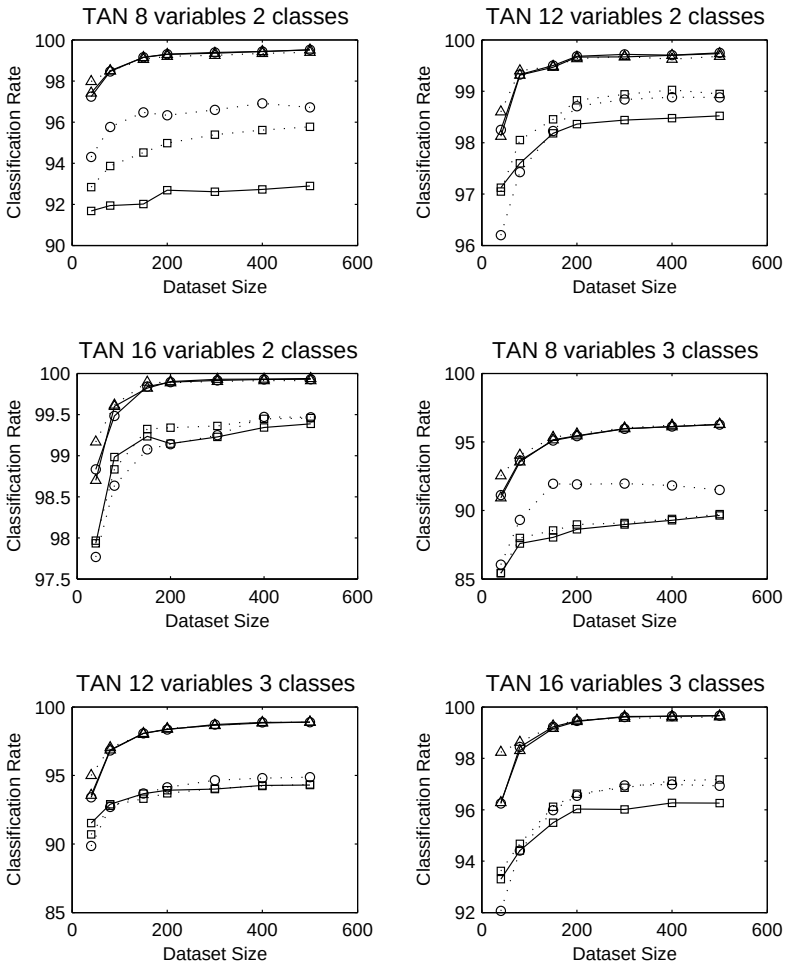


Fig. 5.8. Experiments with datasets sampled from random TAN models. Solid lines represent generative learning while dotted lines discriminative learning. □ denotes NB models; △ TAN models and ∘ cTAN models. Each point in the plot represents the classification rate on average over the 500 different datasets.

likelihood and the classification rate for each model (NB disc., TAN disc. and cTAN disc). In order to measure the correlation between the log-likelihood, conditional log-likelihood and the classification rate we split the results from the ten thousand models with random parameters into smaller subsets with one hundred models. Thus, the figures reported in Table 5.7 are the values of the Spearman's correlation coefficient on average over all the subsets for each model (naive Bayes, TAN and cTAN) so that the correlation coefficient is measured between the log-likelihood and the classification rate (generative models -denoted in Table 5.7 as gen.-) and between conditional log-likelihood

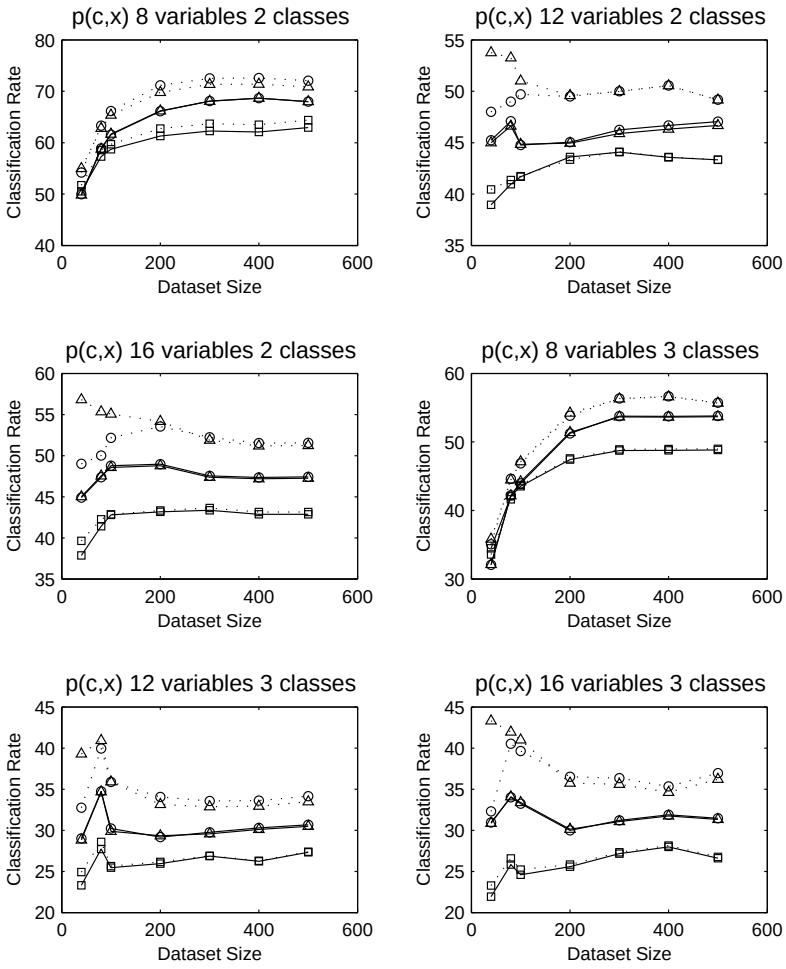


Fig. 5.9. Experiments with datasets sampled from random joint probability distributions. Solid lines represent generative learning while dotted lines discriminative learning. □ denotes NB models; △ TAN models and ∘ cTAN models. Each point in the plot represents the classification rate on average over the 500 different datasets.

and the classification rate (discriminative models -denoted in Table 5.7 as disc.-). This methodology allows to perform a Wilcoxon test in order to see whether or not the differences between the Spearman's correlation coefficient for the generative (NB gen., TAN gen. and cTAN gen.) and discriminative models (NB disc., TAN disc. and cTAN disc.) are statistically significant. The results from the test are not reported in Table 5.7 because the differences between the Spearman's correlation coefficient for the generative and discriminative models are always, in these experiments, statistically significant at the 1% level.

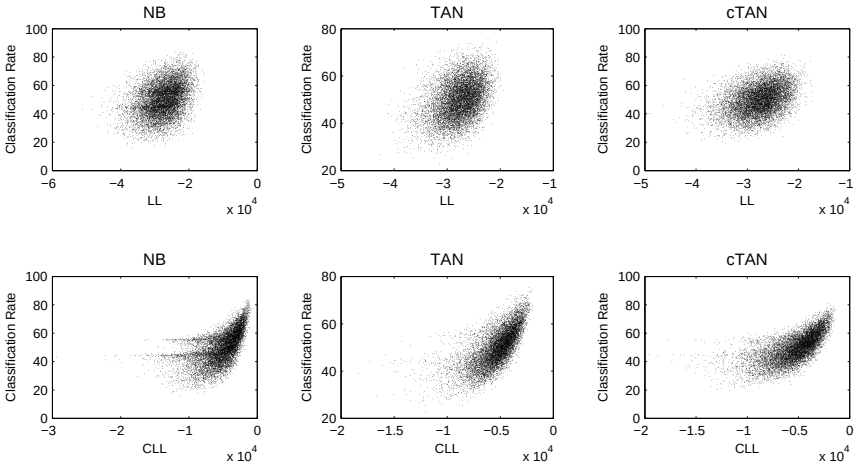


Fig. 5.10. Plot of the relation between LL, CLL and classification rate for `Australian` dataset.

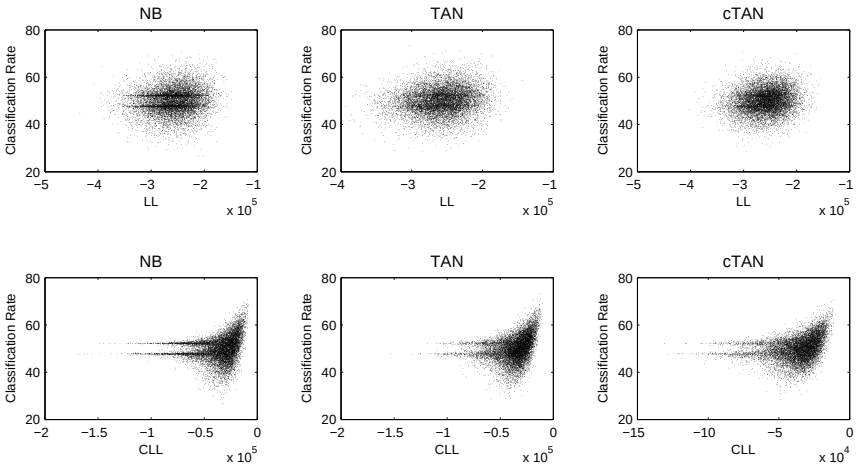


Fig. 5.11. Plot of the relation between LL, CLL and classification rate for `Chess` dataset.

In the experiments with UCI datasets we can observe three different tendencies in the relationship between the log-likelihood, conditional log-likelihood and classification rate. In Figures 5.10 (`Australian` dataset), 5.11 (`Chess` dataset) and 5.12 (`German` dataset) it can be seen that the log-likelihood score is not very related to the classification rate. In fact, the values of the Spearman's correlation coefficient (see Table 5.7) for the generative models in `Australian`, `Chess` and `German` datasets are very low. By contrast, although the values of Spearman's correlation coefficient for the discriminative models in `Australian`, `Chess` and `German` datasets are also low, the

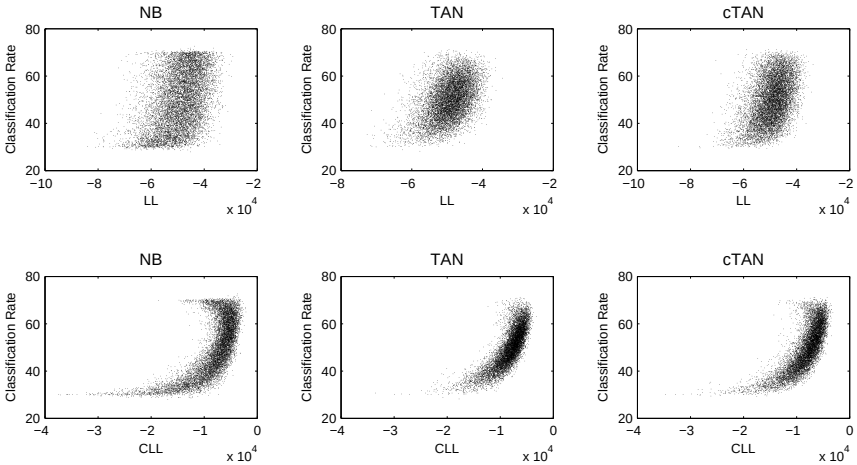


Fig. 5.12. Plot of the relation between LL, CLL and classification rate for `German` dataset.

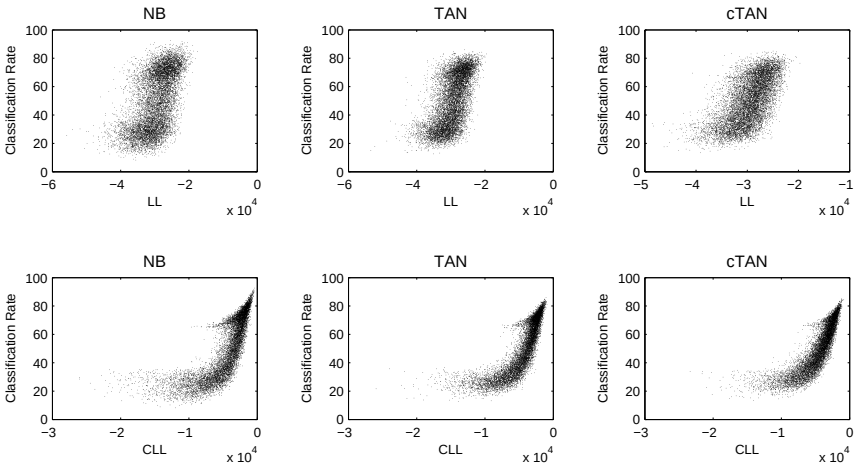


Fig. 5.13. Plot of the relation between LL, CLL and classification rate for `Breast` dataset.

correlation between the conditional log-likelihood and the classification rate is stronger than the correlation between the log-likelihood and the classification rate. On the other hand, there are some datasets such as `Breast` (Figure 5.13), `Hepatitis` (Figure 5.14) and `Lymphography` (Figure 5.15) where we can observe some correlation between the log-likelihood and the classification rate . Nevertheless, the correlation between the conditional log-likelihood and classification rate, in these datasets, is clearly stronger than the correlation between the log-likelihood and classification rate (see also Table 5.7). Finally, in `Flare` dataset (Figure 5.16) we can observe a strong relationship

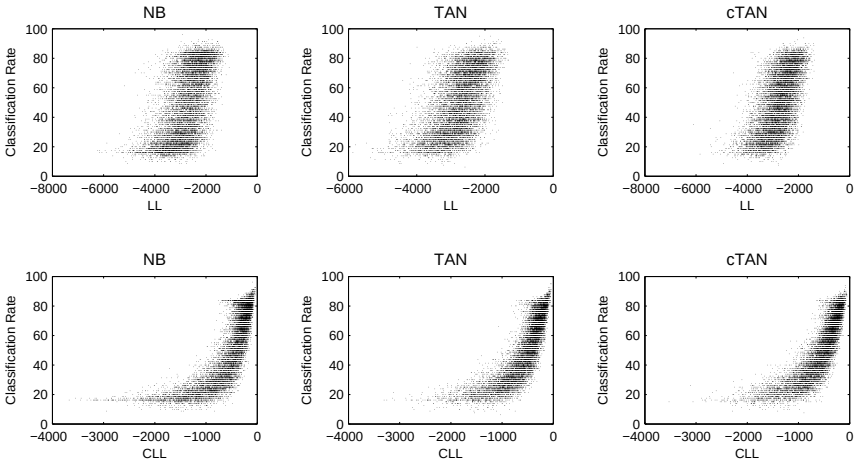


Fig. 5.14. Plot of the relation between LL, CLL and classification rate for `Hepatitis` dataset.

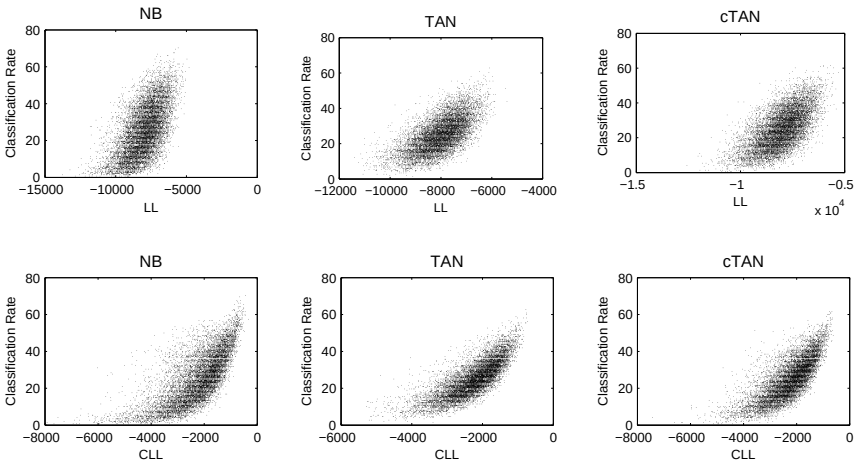


Fig. 5.15. Plot of the relation between LL, CLL and classification rate for `Lymphography` dataset.

between the log-likelihood and the classification rate. However, in this dataset there is an even stronger relationship between the conditional log-likelihood and the classification rate. Hence, we can conclude that, although it depends on each specific dataset, in general the conditional log-likelihood and therefore discriminative learning seems a better approach to learn the parameters of Bayesian network classifiers when we want to maximize the classification rate. Nevertheless, the approach to the discriminative learning of the structure of TAN models (cTAN) proposed in this dissertation does not seem to

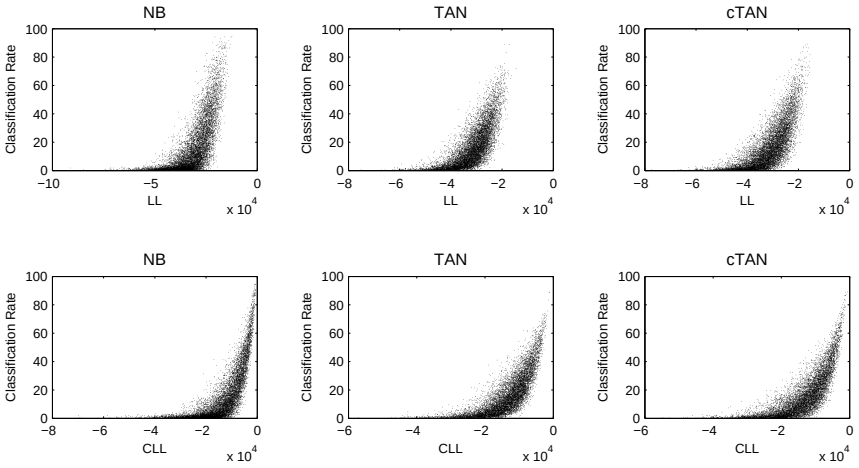


Fig. 5.16. Plot of the relation between LL, CLL and classification rate for `Flare` dataset.

	NB gen.	NB disc.	TAN gen.	TAN disc.	cTAN gen.	cTAN disc.
`Australian`	0.3394	0.6690	0.3743	0.7291	0.3732	0.7193
`Chess`	0.1061	0.3117	0.1377	0.4008	0.1446	0.3828
`German`	0.3802	0.6908	0.4313	0.8015	0.4111	0.7861
`Breast`	0.5734	0.8780	0.6162	0.9047	0.6288	0.9048
`Hepatitis`	0.5819	0.8690	0.5913	0.8802	0.6059	0.8897
`Lymphography`	0.5916	0.7447	0.6428	0.8009	0.6236	0.7712
`Flare`	0.8024	0.8733	0.7955	0.8515	0.7988	0.8554

Table 5.7. Spearman's correlation coefficient between LL and classification rate and between CLL and classification rate for the experiment with UCI datasets.

contribute to obtain a better classification rate than the generative leaning (TAN models).

5.5 Conclusions and Future Work

Bayesian network classifiers are usually considered generative classifiers because the learning process for these classifiers usually attempts to maximize the log-likelihood function. However, there is a growing interest in the discriminative learning of Bayesian network classifiers. In this chapter we have overviewed some approaches for the discriminative learning for both structure and parameters for Bayesian network classifiers. Additionally, we have presented the TM algorithm to learn the parameters and the discriminative-construct-TAN and structural TM algorithms to learn the structure by maximizing the conditional log-likelihood. The experiments carried out in this

chapter show that the discriminative learning of parameters is usually a good choice to learn Bayesian network classifiers, especially when the structure of the model is not able to capture all the relationships among variables which are present in the dataset. This is a usual situation, for instance, when simple models such as naive Bayes or TAN are used in real problems. Nevertheless, the generative learning is much more efficient than discriminative learning and because of this efficiency, generative learning is sometimes preferred.

In the case of discriminative learning of structures, in the experiments, the proposed methods do not perform as well as the discriminative learning of the parameters. In general, a model with a generative structure and discriminative parameters perform better than the model whose structure is learned by a discriminative method. The discriminative-construct-TAN algorithm used in the experiments presents two main disadvantages. First, we have decided to maintain the TAN structure, this can lead to include arcs that decrease the conditional log-likelihood score in the model. Second, the algorithm is a greedy approach, and therefore the obtained structure may not be the best one. This can be corrected by using heuristic search methods to obtain the discriminative structure. On the other hand, the structural TM algorithm relies on the conditional BIC (cBIC) metric. It is known that the use of cBIC (and also BIC) metric tends to select simple models because of the penalization term, especially in the case of cBIC score, this penalization term may dominate the conditional log-likelihood. It is interesting to investigate other metrics to guide the searching process. Moreover, the TM algorithm only learns the parameters for Bayesian network classifiers up to TAN complexity (when the variables are multinomial). This algorithm can be extended to more complex models in order to allow a more complex structural search with the structural TM algorithm. More future work may include the application of the discriminative learning to real problems. In order to use the discriminative learning of Bayesian network classifiers in real problems, it should be very interesting to study the characteristics of the problems where the discriminative learning performs better than generative learning. Recently, Long *et al.* (2006) present a problem which can be solved by a discriminative approach but not by a generative approach. It is very interesting to extend this idea to a more general context in order to analyze the limitations of discriminative and generative learning to solve specific problems from a theoretical approach.

Part III

Clustering

6

Data Clustering

Data clustering can be defined as the process of partitioning a dataset into subgroups or clusters so that elements in the same partition share, ideally, some common characteristics. That is, the clustering process aims for obtaining the unknown underlying group-structure of the data.

In this chapter we introduce the data clustering problem and we classify clustering algorithms into crisp or fuzzy algorithms. We briefly overview both clustering approaches by presenting a representative algorithm for each approach. Finally, we focus our attention on probabilistic clustering which can be seen as a special type of fuzzy clustering and more specifically we focus on Bayesian network models for data clustering.

6.1 Clustering Data

A clustering algorithm aims to retrieve a group structure from a dataset of N unlabeled instances $D = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$ where each data instance is characterized by n predictive variables. We say that the dataset D is composed of unlabeled instances because the presence of an additional hidden random variable C is assumed, named class or cluster variable, which defines the cluster membership. Therefore, the dataset D can also be denoted as $D = \{(c^{(1)}, \boldsymbol{x}^{(1)}), \ldots, (c^{(N)}, \boldsymbol{x}^{(N)})\}$ where the values of C are unknown. Because of the existence of this hidden random variable, the clustering problem can be seen as learning from incomplete or missing data. However, these terms are more general as they only mean that some of the instances of the dataset are not fully specified, but they do not refer to the presence of the hidden clustering variable. A more correct term to denote data clustering is learning from unlabeled data or simply unsupervised learning.

Usually, the number of clusters underlying the dataset, denoted in this dissertation as r_C, is not known and the clustering process may involve the identification of the number of groups. This is a difficult task since normally there is not a single solution for the number of clusters underlying the dataset.

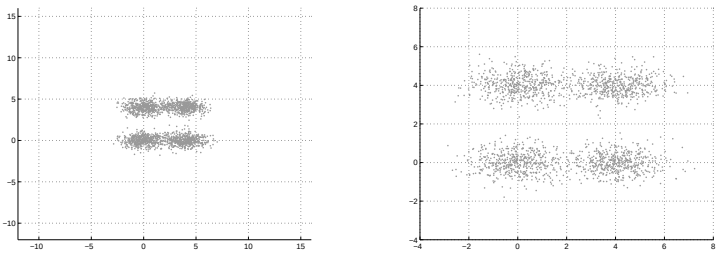


Fig. 6.1. Plot of the elements in a dataset using two different scales.

Even for the same dataset, there can be several answers depending on the scale or granularity of interest. For instance, Figure 6.1 shows a graphical example where one would assert that there are two clusters in the figure on the left and four clusters in the figure on the right, but both figures actually correspond to the same dataset plotted in different scales.

In the case where the number of clusters is unknown, there are many proposals to overcome the problem. One approach involves the use of a data preprocessing step to determine the most likely r_C for the current data (Climescu-Haulica, 2006). On the other hand, another approach to identify the number of clusters uses different values of r_C to induce clustering models for the given dataset. Then the clustering partitions or clustering models with different number of clusters are compared in order to determine the most convenient r_C value *a posteriori*. Some methods to determine the most convenient number of clusters are, for instance, likelihood-ratios (Binder, 1978; Everitt, 1980; Binder, 1981), Bayes factors (Dasgupta and Raftery, 1998; Fraley and Raftery, 1998; Gangnon and Clayton, 2007) or evolutionary algorithms (Bandyopadhyay, 2005; Wei and Traore, 2005). Another well-used method to determine the number of clusters is the elbow criterion which is a subjective method that can be based on any particular clustering quality measure. This method aims to identify when the use of more clusters does not yield a relevant improvement in the quality of the obtained clustering partition.

As one may expect, not all the partitions of D into r_C clusters are equally desirable. While some are plausible representations of the underlying group structure of the dataset D, other partitions inform us poorly about this structure. Therefore, data clustering can be redefined as finding the best description of r_C clusters for a dataset D according to some metric, score or criterion, known as the clustering criterion. Hence, data clustering is equivalent to an optimization problem where the function to be optimized is the clustering criterion and the best solution is searched in the space of all the possible data partitions with r_C clusters.

According to the characteristics of the different clustering methods, they can be classified in different categories (Jain *et al.*, 1999). In this dissertation,

we make a distinction between hard, fuzzy and probabilistic clustering. In the following sections we briefly overview these clustering approaches as well as some clustering methods belonging to each approach.

6.1.1 Hard Clustering

Hard clustering algorithms, also named as crisp clustering algorithms, obtain a clustering partition where each data is assigned to exactly one cluster. Therefore, the clustering process is reduced to complete each data instance in D with the label of the cluster which it belongs to. However, this is not an easy task because of the high number of possible partitions of D into r_C clusters, which is given by the Stirling number of the second kind:

$$\frac{1}{r_C!}\sum_{j=1}^{r_C}(-1)^{r_C-j}\binom{r_C}{j}j^N \tag{6.1}$$

This quantity grows exponentially with N and r_C. In fact, the partition of a dataset into r_C clusters is known to be an NP-hard problem (Garey and Johnson, 1979). Therefore, heuristic methods should be used to solve the data clustering problem.

Hard clustering algorithms can also be divided in two different categories: hierarchical and partitional algorithms. Hierarchical methods do not produce a single data partition but they obtain a dendrogram which represents nested clusters and the similarity level of the nested clusters (see Figure 6.2). The similarity between elements and how the linkage between elements is performed in order to form the clusters are parameters that differ among hierarchical clustering approaches. For instance, the single-link approach using Euclidean distance is a very popular choice (Sneath and Sokal, 1973; Jain and Dubes, 1988). Hierarquical clustering algorithms obtain the nested clustering in N iterations. In the first iteration, each data instance is representative or centroid of a cluster. Then, the two nearest clusters, according to the selected distance, are grouped together so that they act as a single cluster for the next iteration. Finally, the last iteration of the algorithm obtains a cluster that groups all the other clusters obtained throughout the iterations. Note that, although the dendrogram represents nested clusters, we can obtain crisp partitions with a different number of clusters (from N to 2) by cutting the tree on a specific level.

On the other hand, partitional methods describe each cluster by means of a subset of instances from D so that each instance can only belong to a single cluster. Therefore, the partitional clustering obtains a single data partition. A well-known heuristic algorithm to deal with partitional clustering problems is the K-means algorithm (Forgy, 1965; MacQueen, 1967; Andenberg, 1973; Hartigan, 1975; Jain and Dubes, 1988; Fukunaga, 1990). The K-means algorithm finds a locally optimal partition of the dataset D into K clusters

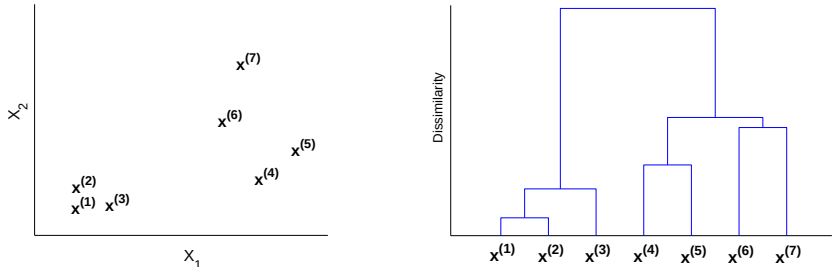


Fig. 6.2. Hierarchical clustering. Plot of the elements in the dataset (on the left) and dendrogram obtained by a hierarchical clustering algorithm (on the right).

according to the square-error criterion[1]. Note that K, in the K-means algorithm represents the number of clusters but in this dissertation we denote it as r_C. The K-means is an iterative algorithm. Initially, a clustering partition of the dataset is obtained, for instance, at random and the centroids of the clusters are calculated. At each iteration, the K-means algorithm reallocates the data instances by assigning them to the closest cluster in terms of the distance between the data sample and the cluster centroid. Then, the centroids of the clusters are re-calculated. The algorithm converges when the centroids of the clusters do not vary in two consecutive iterations. Although the K-means algorithm has a good performance, specially in large dataset, the algorithm is very sensitive to the initial partition (Milligan, 1980; Meila and Heckerman, 1998; Peña *et al.*, 1999a).

6.1.2 Fuzzy and Probabilistic Clustering

Hard clustering approaches generate partitions so that each data instance belongs to one and only one cluster. Therefore, the obtained clusters are disjoint. Fuzzy clustering extends clustering methodology by associating each data instance with every cluster using a membership function (Zadeh, 1965). Therefore, these algorithms yield a clustering but not exactly a data partition. One of the well-known fuzzy clustering methods is the fuzzy c-means (Bezdek, 1981) which is an extension of the K-means method to deal with fuzzy clusters.

Probabilistic clustering is another type of non-crisp clustering. Probabilistic clustering is a model-based approach where it is aimed to describe the physic mechanism that generated the dataset by means of probability distributions (Figure 6.3). The most used approach for probabilistic clustering is based on the theory of finite mixture models (Duda and Hart, 1973; Demp-

[1] The sum of the distance between each instance of D and its closest cluster centroid.

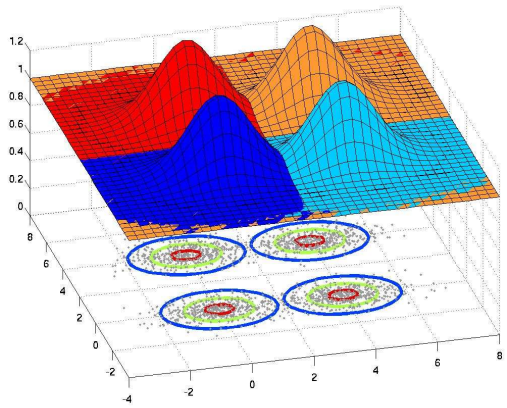


Fig. 6.3. Probabilistic clustering. In this graphical example, each clustering is represented by a probability density function.

ster *et al.*, 1977; McLachlan and Basford, 1989; Banfield and Raftery, 1993; McLachlan and Peel, 2000). Thus, the clustering model is written as:

$$\rho(\boldsymbol{x}) = \sum_{j=1}^{r_C} p(c^j)\rho(\boldsymbol{x}|c^j) \tag{6.2}$$

where $p(c^j)$ is the probability of selecting the physical process associated to the j-th cluster and $\rho(\boldsymbol{x}|c^j)$ is the joint generalized density function which describes the physical process associated to the j-th cluster. Therefore, probabilistic clustering aims to find the best parameter set for the model in Equation 6.2 according to some clustering criterion. Usually, this clustering criterion is a measure of how distant the learned model and the *true*[2] probability distributions are. Unfortunately, the true probability distributions are not usually known. Then, the dataset D is normally used to measure how well the learned model describes it. That is, the data clustering criterion is the likelihood of the dataset D given the model parameters $\boldsymbol{\theta} = (\theta_{C-1}, \ldots, \theta_{C-r_C}, \boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^{r_C})$. This likelihood function can be expressed in its logarithmic form as:

$$LL = \sum_{d=1}^{N} \log \sum_{j=1}^{r_C} p(c^j|\theta_{C-j})\rho(\boldsymbol{x}^{(d)}|c^j, \boldsymbol{\theta}^j) \tag{6.3}$$

Given the log-likelihood as clustering criterion, the probabilistic clustering problem reduces to obtaining the parameters of the finite mixture model,

[2] The *true* probability distributions are the probability distributions that model the physical mechanism which generated the dataset D.

$\boldsymbol{\theta} = (\theta_{C-1}, \ldots, \theta_{C-r_C}, \boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^{r_C})$, that maximize Equation 6.3. However, this is not an easy task because of the high number of parameters in the model and the large search space. Hence, heuristic search strategies to learn the set of parameters for the clustering model are usually needed. One of the most popular heuristic searching methods is the expectation-maximization (EM) algorithm (Dempster *et al.*, 1977; McLachlan and Krishnan, 1997). This algorithm is used in the data clustering experiments performed in this dissertation and is introduced in following sections.

In spite of the computational complexity of learning probabilistic clustering models, they are very popular because they are statistically well founded in contrast to partitional clustering methods which sometimes suffer from a lack of theoretical basis. In fact, there are several authors who claim that probability clustering is the only completely satisfactory due its well-defined theoretical foundations (Duda and Hart, 1973; Aitkin *et al.*, 1981; McLachlan and Peel, 2000).

The part of this dissertation devoted to data clustering is concerned with probabilistic data clustering. Specifically, the clustering model, which is given by the joint probability distribution of the random variable $(C, \boldsymbol{X})$, is induced from D by means of unsupervised learning of Bayesian networks. The following section briefly present Bayesian network models used for clustering purposes.

6.2 Bayesian Networks for Clustering

Due to the high number of parameters needed to define a joint probability distribution, probabilistic clustering may sometimes be inefficient. However, these probability distributions can be represented by probabilistic graphical models in general and by Bayesian networks in particular (see Chapter 2). Then, the representation of the joint probability distribution can take advantage of the factorization given by the probabilistic graphical models which is derived from the conditional (in)dependence relationships among the variables. Although, depending on the type of graph used to represent the relationships among variables and the parametric form of the generalized joint probability distribution, different probabilistic graphical models can be used for data clustering, in this dissertation we are concerned only with Bayesian networks models.

Bayesian network models for clustering are identical to the Bayesian networks used for supervised classification (see Chapter 4) except for the fact that they take into account the presence of the hidden cluster variable. Therefore, data clustering using Bayesian network is a specific case of learning Bayesian network from incomplete or missing data. In the case of complete dataset, the structure is learned by maximizing a score metric usually related to the log-likelihood function such as MDL metric. Even when the complete dataset allows an efficient calculation of the log-likelihood function, the learning of the Bayesian network structure is an NP-Hard problem (Chickering *et al.*, 1994)

because of the huge number of structures. This situation is aggravated in the case of data clustering because the presence of a hidden variable prevents the decomposition of the log-likelihood function making the evaluation of the score much more inefficient. Therefore, several methods to learn Bayesian networks for clustering neglect the structural search by requiring a fixed network structure. This fixed network structure can be provided by an expert or simply by using a simple Bayesian network model for clustering such as a naive Bayes or a selective naive Bayes (see Chapter 4). On the other hand, other methods propose a search in the joint space of parameters and structures in order to learn the best Bayesian network model for clustering. In following the sections we present these two approaches. The EM (Dempster *et al.*, 1977) is an algorithm that can be used to learn the parameter of a Bayesian network model assuming that the structure of the model is already known. By contrast, the structural EM (Friedman, 1997) learn both the structure and the parameters of a Bayesian network model for clustering. Note that, as well as in Chapter 2, we firstly introduce the parameter learning because some aspects of the EM algorithm are used for learning the structure of the model with the structural EM algorithm.

6.2.1 Parameter Learning: EM Algorithm

The Expectation Maximization (EM) algorithm is a general framework to learn the parameters of a model by maximizing the likelihood function in the presence of missing data. It is essentially an iterative optimization algorithm which, at least under certain conditions, converges to parameter values at a local maximum of the likelihood function. One of the earliest works related to the EM algorithm is Hartley (1958), but the work where the EM algorithm is proposed and its convergence properties are studied is presented in Dempster *et al.* (1977). Other popular and useful references for the EM algorithm are Tanner (1996) and McLachlan and Krishnan (1997).

In this section, we present the EM algorithm as a method to learn the parameters of Bayesian network models for clustering. Hence, the EM algorithm only learns the parameter set of the model and the structure is assumed to be already known. The idea behind the EM algorithm is simple: at each step t, the algorithm alternates between estimating the parameters of the model, $\boldsymbol{\theta}^{(t)}$, and the values of the hidden variable, C. However, instead of finding the best values of C given the current estimation of the parameters, the algorithm computes the probability distribution of C, $p(c|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})$. These calculations are performed in the two steps of the algorithm: Expectation (E step) and Maximization (M step).

- E Step: Intuitively, we can see this step as a *completion* of the values for the cluster variable, which are missing. This step obtains the expected log-likelihood function (Equation 6.4) which is maximized in the M step.

$$Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)}) = E_{\boldsymbol{\theta}^{(t)}}[\log p(D|S,\boldsymbol{\theta})|D]$$
$$= \sum_{d=1}^{N} \log \sum_{c} p(c|\boldsymbol{x}^{(d)}, S, \boldsymbol{\theta}^{(t)}) p(c, \boldsymbol{x}^{(d)}|S, \boldsymbol{\theta}) \tag{6.4}$$

In the case of Bayesian network models, the E step involves the calculation of the expected sufficient statistics of the dataset, $E(N_{ijk}|\boldsymbol{\theta}^{(t)}, S)$ for each predictive variable X_i and parent set $\boldsymbol{Pa}_i$ with $i = 1, \ldots, n$; $j = 1, \ldots, q_i$; and $k = 1, \ldots, r_i$.

- *M* Step: The M step obtains a new set of parameters for the model by maximizing the expected log-likelihood function obtained in the E step:

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)}) \tag{6.5}$$

In the case of Bayesian network models, the new parameters of the model can be calculated by using the expected sufficient statistics previously obtained in the E step as if they were actual sufficient statistics from a complete dataset:

$$\theta_{ijk} = \frac{E(N_{ijk}|\boldsymbol{\theta}^{(t)}, S)}{E(N_{ij}|\boldsymbol{\theta}^{(t)}, S)} \tag{6.6}$$

where $E(N_{ij}|\boldsymbol{\theta}^{(t)}, S) = \sum_{k=1}^{r_i} E(N_{ijk}|\boldsymbol{\theta}^{(t)}, S)$

Alternatively, the EM algorithm can be used to learn the *maximum a posteriori* parameters instead of the *maximum likelihood* ones. Thus, we are able to include *prior* knowledge about the parameters in the learning process. Nevertheless, even if no *prior* information about the parameters is known, the EM algorithm can be used to obtain *maximum a posteriori* parameters by using non-informative *priors*. Thus, the MAP parameters should be calculated in the M step as:

$$\theta_{ijk} = \frac{E(N_{ijk}|\boldsymbol{\theta}^{(t)}, S) + \alpha_{ijk}}{E(N_{ij}|\boldsymbol{\theta}^{(t)}, S) + \alpha_{ij}} \tag{6.7}$$

where $(\alpha_{ij1}, \ldots, \alpha_{ijr_i})$ are the hyperparameters for the *prior* probability distribution of the parameters of the model, which is normally a Dirichlet probability distribution $(\theta_{ij1}, \ldots, \theta_{ijr_i}) \backsim D(\theta_{ij1}, \ldots, \theta_{ijr_i}|\alpha_{ij1}, \ldots, \alpha_{ijr_i})$, and being $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$.

Both expectation and maximization steps of the EM algorithm are successively repeated until the algorithm converges to a local maximum. That is, the algorithm obtains the same value of the log-likelihood function in two consecutive iterations of the algorithm. However, in practice, it is usually said that

the algorithm has converged when the difference between the log-likelihood values obtained in two consecutive iterations of the algorithm is smaller than certain threshold fixed in advance.

Due to the popularity of the EM algorithm to learn probabilistic models from data with missing values, there are many proposals to improve the algorithm, for example, by accelerating the convergence to a local maximum. Although there are many others, some of these proposals are reviewed, for instance, in McLachlan and Krishnan (1997).

6.2.2 Learning Structure and Parameters: Structural EM Algorithm

The structure of the Bayesian network model for clustering can be learned by combining the EM algorithm with a structural search. This method requires the EM algorithm to learn the parameters of each candidate structure in order to evaluate the goodness of the candidate model. This process is extremely inefficient and it can only be used in problems where there are only a few choices in the network structure. For instance, Cheeseman and Stutz (1996) and Chickering and Heckerman (1997) select the number of values for a single hidden variable in networks with a fixed structure, and Heckerman (1995) describes an experiment with a single missing value and five observable variables.

A more efficient method to learn Bayesian network models for clustering is the structural EM. The structural EM algorithm extends the EM algorithm in order to search the *maximum a posteriori* model in the joint space of parameters and structures (Friedman, 1997). Although in the original paper (Friedman, 1997) the author named the algorithm model selection EM, another more recent work by the same author (Friedman, 1998) refers to the algorithm as structural EM. At each iteration, the structural EM can either find better parameters for the current structure, or select a new structure. The former case follows the standard approach performed by the EM algorithm (Section 6.2.1), while the latter case is a structural search. This structural search obtains the structure which maximizes the expected MDL score given the model from the previous iteration of the algorithm. Therefore, if a Bayesian network model is represented as $\mathcal{B} = (S, \boldsymbol{\theta})$ and $\mathcal{B}^{(t)}$ represents the model obtained in the t-th iteration of the algorithm, the function maximized by the structural EM algorithm is:

$$
\begin{aligned}
Q(\mathcal{B}; \mathcal{B}^{(t)}) &= E_{\mathcal{B}^{(t)}}\left[\log p(D|\mathcal{B}) - \frac{1}{2}\log N dim(S)|D\right] \qquad (6.8)\\
&= \sum_{d=1}^{N}\sum_{c} p(c|\boldsymbol{x}^{(t)}, \mathcal{B}^{(t)})\left(\log p(c, \boldsymbol{x}^{(t)}|\mathcal{B}) - \frac{1}{2}\log N dim(S)\right)
\end{aligned}
$$

Choose $S^{(0)}$ and $\boldsymbol{\theta}^{(0)}$ at random

for $t = 0, 1 \ldots$ until convergence **do**

Run EM to learn $\boldsymbol{\theta}^{(t)}$ for a fix structure $S^{(t)}$

– Search for the best structure –

Find the structure $S^{(t+1)}$ that maximizes $Q(\mathcal{B}; \mathcal{B}^{(t)})$

– Obtain the parameters for $S^{(t+1)}$ –

$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} = Q(S^{(t+1)}, \boldsymbol{\theta}; S^{(t)}, \boldsymbol{\theta}^{(t)})$

end for

Fig. 6.4. General pseudo-code for the structural EM algorithm (Friedman, 1997).

where $dim(S)$ is the dimension of the structure S which is usually given by the number of parameters. Figure 6.4 shows the pseudo-code for the structural EM algorithm.

The structural EM algorithm learns the Bayesian network model by using the MDL as a score for the search of the model structure which can be seen as an approximation to the marginal log-likelihood function. Friedman (1998) proposes the Bayesian structural EM algorithm which, instead of using the MDL score, directly maximizes the marginal log-likelihood when learning the structure of the model. The use of the marginal log-likelihood instead of the MDL score is, from a Bayesian point of view, a better criterion to guide the structural search. Alternatively, Peña *et al.* (2000) propose a modification of the Bayesian structural EM by using an improved EM algorithm, the BC+EM (Peña *et al.*, 1999b).

7

Bayesian Model Averaging

Standard approaches in machine learning and statistical analysis often ignore uncertainty about model selection. It is a common praxis to select a single model and proceed as if the selected model had generated the data. However, this approach neglects the uncertainty in model selection and leads to overconfident decisions, that is, the decisions based on that model are more risky than one thinks they are. In this chapter we review Bayesian model averaging, which is a technique that accounts for the uncertainty inherent in the model selection process by averaging over many different competing models. Moreover, we present novel approaches to deal with Bayesian model averaging for data clustering problems.

7.1 Introduction to Bayesian Model Averaging

From a data-driven approach, the information about a problem is obtained from a dataset and this information is used to learn a model. Then, the model is used to make predictions (in the case of supervised classification problems) or to obtain a description of the underlying group-structure of the data (in the case of data clustering problems). Unfortunately, information about the model that generated the data is not usually available. Hence, even when the model is learned from the dataset and it fits the data reasonably well (or even it is the best model fitting the data), there is no guarantee that the learned model is the *true* model, that is, the model that had generated the dataset.

Let assume that we are somehow provided with a dataset D and we use a leaning algorithm to obtain a model $\mathcal{M}$ from D. This model fits the data reasonably well. However, let suppose that there exists an alternative model $\mathcal{M}'$ which also provides a good fitting to the dataset. Why should we choose $\mathcal{M}$ instead of $\mathcal{M}'$? The uncertainty associated to the finite dataset may result in the fact that $\mathcal{M}$ fits the dataset better than $\mathcal{M}'$ even when $\mathcal{M}'$ is closer to the *true* model that generated the data. Moreover, the fewer samples the dataset has, the higher the uncertainty in model selection is. Therefore, basing

inferences only on $\mathcal{M}$ is risky (Dijkstra, 1988). Bayesian model averaging (Leamer, 1978; Madigan and Raftery, 1994; Chatfield, 1995; Hoeting *et al.*, 1999) provides a methodology to deal with this problem. This methodology describes the dataset by means of the posterior probability distribution, which is obtained by averaging over the posterior distributions of the considered models:

$$p(c, \boldsymbol{x}|D) = \sum_{\mathcal{M} \in \boldsymbol{\mathcal{M}}} p(c, \boldsymbol{x}|\mathcal{M}, D) p(\mathcal{M}|D) \tag{7.1}$$

where $\boldsymbol{\mathcal{M}}$ is the class of models considered for the averaging.

Taking into account Bayesian network models, Bayesian model averaging has to deal with uncertainty in the selection of the structure and the parameters:

$$p(c, \boldsymbol{x}|D) = \sum_{S} \int p(c, \boldsymbol{x}|S, \boldsymbol{\theta}) p(\boldsymbol{\theta}|S, D) d\boldsymbol{\theta}\ p(S|D) \tag{7.2}$$

where the posterior probability of the structure S is given by:

$$p(S|D) = \frac{p(D|S)p(S)}{\sum_{S'} p(D|S')p(S')} \tag{7.3}$$

and the marginal likelihood given S is:

$$p(D|S) = \int p(D|\mathcal{M}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|S) d\boldsymbol{\theta} \tag{7.4}$$

Unfortunately, although Bayesian model averaging is an attractive solution that can deal with uncertainty in both parameters and structure selection, it presents several difficulties. On the one hand, the huge number of terms in the summation over the structures (see Equation 7.2) makes Bayesian model averaging unfeasible. This problem can be overcome by averaging over a reduced class of models. However, this solution includes uncertainty in the Bayesian model averaging process and then, the selection of a proper class of models may be a crucial task. On the other hand, the integrals over the parameters (see Equations 7.2 and 7.4) are sometimes very hard to compute. Due to these drawbacks, it is usually unfeasible an exact Bayesian model averaging calculation and therefore, several approximations are proposed. In the following section we present some methods to approximate Bayesian model averaging of Bayesian network models. Finally, we present a new algorithm to approximate Bayesian model averaging of naive Bayes and TAN models for clustering problems.

Note that, as the work presented in this dissertation is focused on supervised classification and clustering, the formulas for Bayesian model averaging presented here are adapted to these problems by distinguishing between the predictive variables $\boldsymbol{X}$ and the class or cluster variable C. Nonetheless, although in this chapter we review some proposals for the Bayesian model averaging of Bayesian networks for supervised classification problems, we focus the development of new algorithms on data clustering problems.

7.2 Bayesian Model Averaging of Bayesian Networks

Probabilistic graphical models in general and Bayesian networks in particular have received the attention of Bayesian model averaging community. However, the majority of the approximated methods for Bayesian model averaging are proposed for supervised classification tasks, where the dataset does not contain missing values. For instance, Madigan and Raftery (1994) propose the Occam's window method which simplifies model averaging problem by averaging over a set of parsimonious data-supported models. This set of models is selected so that models which predict the data much worse than the best model are not taken into account. Additionally, those complex models which receive less support from the data than their simpler counterpart are also excluded from the set of selected models. An alternative approach is presented by Madigan *et al.* (1994) and Madigan and York (1995) where Markov chain Monte Carlo simulation (Metropolis and Ulam, 1949; Gilks *et al.*, 1998) is used to approximate Bayesian model averaging calculations. Related approaches have also been adopted by other researchers such as Giudici and Green (1999) who propose a Markov chain Monte Carlo approach over the class of junction trees undirected graphical models that are decomposable.

In Bayesian model averaging of Bayesian network models the theoretical developments presented in Buntine (1991) and Friedman and Koller (2003) are of special interest. In these works they propose a special decomposition which allows to obtain the posterior probability of the arcs in a network. This idea has been applied to learn Bayesian network models for classification: Dash and Cooper (2002) and Cerquides and López de Mántaras (2003a) present exact Bayesian model averaging calculations over the set of selective naive Bayes models; Dash and Cooper (2003) and Cerquides and López de Mántaras (2003b, 2005) perform exact model averaging over TAN models given an ancestral ordering among the variables; Dash and Cooper (2004) extend model averaging calculations to more complex Bayesian networks for a given ordering among the variables; and Hwang and Zhang (2005) relax the selection of a variable ordering by averaging over multiple node orders. Alternatively, Meila and Jaakkola (2000, 2006) present a family of decomposable *priors* over structure and parameters of tree Bayesian networks for which Bayesian model averaging calculations with complete observations are tractable.

Although Bayesian model averaging has shown to perform well on classification problems and is theoretically an optimal method to deal with uncertainty in model selection, Domingos (2000) argues that Bayesian model averaging can actually exacerbate the over-fitting problem in machine learning. Moreover, he shows how more *ad hoc* techniques for model combination such as bagging can overcome Bayesian model averaging. This arguments are questioned by Minka (2002) who shows that Bayesian model averaging is not a technique for model combination but a technique for soft model selection.

Unfortunately, for clustering problems, the integrals from Equation 7.2 and 7.4 can not be solved in closed form. This makes the exact Bayesian model averaging calculations for clustering models typically intractable and only approximations are feasible. Some attempts to approach the integrals involved in Bayesian model averaging include the use of stochastic simulation or Laplace's approximation (Chickering *et al.*, 1995). However, stochastic simulation methods tend to be computationally expensive and Laplace's approximation may be quite imprecise. Therefore, Bayesian model averaging for clustering is usually approximated by using only the *maximum a posteriori* (MAP) model, which can be obtained by means of the EM algorithm (see Chapter 6). In fact, this is an approximation to Bayesian model averaging because the posterior probability function of the models is peaked at the MAP model. Consequently, the MAP model is the one which contributes the most to the Bayesian model averaging calculations. Nevertheless, the selection of a single model does not take into account the uncertainty in model selection. Several proposals try to overcome the selection of a single model by combining several Bayesian network models in a mixture model of Bayesian networks (Meila and Jordan, 1998; Thiesson *et al.*, 1998). However, mixtures of Bayesian network models can be used to combine several Bayesian networks in a model but they do not address the dependency on a single model as Bayesian model averaging does. Additionally, the Bayesian structural EM algorithm (Chapter 6; Friedman (1998)) learn the structure of a Bayesian network model by maximizing the marginal log-likelihood function. This function can not be solved in closed form for clustering problems, but it can be maximized into the iterations of the structural EM algorithm. Alternatively, Friedman (1998) points out the idea of averaging over a set of models with the highest marginal log-likelihood instead of searching for the best model.

In this dissertation we present a new algorithm named Expectation Model Averaging (EMA) (Santafé *et al.*, 2006b,c) which allows an efficient computation of Bayesian model averaging calculations for clustering problems. In following sections, we present the EMA algorithm for naive Bayes and TAN models as well as an empirical evaluation of the proposed methods.

7.3 Expectation Model Averaging: EMA Algorithm

The expectation model averaging (EMA) is an algorithm to approximate the Bayesian model averaging of Bayesian network models for clustering problems. Although the algorithm can be adapted to average over different classes of Bayesian networks, in this dissertation we only introduce the EMA algorithm for naive Bayes (Santafé *et al.*, 2006b) and TAN models (Santafé *et al.*, 2006c). The EMA algorithm is a variant of the well-known EM algorithm which allows to extend the model averaging calculations proposed by Friedman and Koller (2003), Dash and Cooper (2004) and Cerquides and López de Mántaras (2005) to clustering problems. The EMA algorithm uses the E step of the EM algorithm to deal with the unknown values for the cluster variable. Then, it performs a model averaging step (MA) to obtain $p(c, \boldsymbol{x}|D)$ and thus, the clustering model can be obtained by means of the Bayes rule:

$$p(c^i|\boldsymbol{x}, D) = \frac{p(c^i, \boldsymbol{x}|D)}{\sum_{j=1}^{r_C} p(c^j, \boldsymbol{x}|D)} \tag{7.5}$$

The EMA, as well as the EM algorithm, is an iterative process where the two steps of the algorithm are repeated successively until a stopping criterion is met. At the t-th iteration of the algorithm, a set of parameters $\breve{\boldsymbol{\theta}}^{(t)}$ is calculated. The algorithm stops when the difference between the sets of parameters learned in two consecutive iterations, $\breve{\boldsymbol{\theta}}^{(t)}$ and $\breve{\boldsymbol{\theta}}^{(t+1)}$, is less than threshold ϵ, which is fixed in advance.

In order to use the EMA algorithm, we need to set an initial parameter configuration, $\breve{\boldsymbol{\theta}}^{(0)}$, and the value of ϵ. The values for $\breve{\boldsymbol{\theta}}^{(0)}$ are usually taken at random and ϵ is set at a small positive value.

Note that, although the EMA algorithm can be seen as a general algorithm for Bayesian model averaging over Bayesian network model, this term was originally used to denote the expectation model averaging algorithm of naive Bayes for clustering (Santafé *et al.*, 2006b). Therefore, in order to be coherent with the names given to the algorithms in the works where they were proposed (Santafé *et al.*, 2006b,c), we refer to the expectation model averaging of naive Bayes models for clustering as EMA algorithm and we denote the expectation model averaging of TAN models for clustering as EMA-TAN algorithm.

7.3.1 EMA Algorithm for Naive Bayes

The EMA algorithm for naive Bayes obtains a naive Bayes model for clustering by averaging over all selective naive Bayes structures (see Figure 7.1) where, for each structure, the averaging over parameters is approximated by its MAP configuration. In other words, we obtain a unique naive Bayes model for clustering which is equivalent to the average of the MAP configurations for every selective naive Bayes structure weighted by its posterior probability.

Note that, even though the obtained model is a single naive Bayes model, $\breve{\mathcal{B}} = (\breve{\boldsymbol{\theta}}, S_{nB})$ with S_{nB} being a naive Bayes structure, its parameters, $\breve{\boldsymbol{\theta}}$, are learned taking into account the MAP parameter configuration for every selective naive Bayes structure. Thus, the resulting unsupervised naive Bayes will incorporate into its parameters information about the independence between variables described by the different selective naive Bayes models.

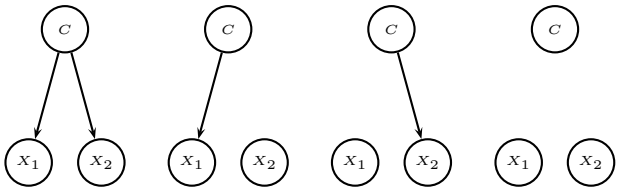


Fig. 7.1. All selective naive Bayes structures with two predictive variables: each predictive variable can be dependent on or independent of C.

Before introducing the Bayesian model averaging calculations related to the EMA algorithm, we need to clarify some notation. Using the classical notation in Bayesian networks already introduced in Chapter 2, the set of parents for variable X_i, with $i = 1, \ldots, n$, is denoted as Pa_i. In this case, for all selective naive Bayes models, $Pa_i \in \{\emptyset, \{C\}\}$. θ_{ijk}, with $k = 1, \ldots, r_i$ and r_i being the number of states for variable X_i, represents the conditional probability of variable X_i taking its k-th value given that Pa_i takes its j-th value. The conditional probability mass function for X_i given the j-th configuration of its parents is designated as $\boldsymbol{\theta}_{ij}$, with $j = 1, \ldots, q_i$, where q_i is the number of different configurations of Pa_i. Finally, $\boldsymbol{\theta}_i = (\boldsymbol{\theta}_{i1}, \ldots, \boldsymbol{\theta}_{iq_i})$ denotes the set of parameters for variable X_i, and $\boldsymbol{\theta} = (\boldsymbol{\theta}_C, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$ represents the whole set of parameters for a selective naive Bayes model, where $\boldsymbol{\theta}_C = (\theta_{C-1}, \ldots, \theta_{C-r_C})$ is the set of parameters for the cluster variable, with r_C the number of clusters fixed in advance.

In order to distinguish between the parameters for different selective naive Bayes models, we introduce the notation θ_{ijk} and θ_{i-k} to denote the parameters when there is an arc between C and X_i, and when there is none respectively. By extension to a general case, we take into consideration the same notation (Q_{ijk} and Q_{i-k}) with any quantity (Q) related to variable X_i.

Finally, we need to make the following five assumptions in order to perform the approximation to Bayesian model averaging:

- *Assumption 1*: Multinomial variables.
  Each variable X_i, with $i = 1, \ldots, n$, is discrete and can take r_i states. The cluster variable is also discrete and, as introduced before, it takes r_C possible states, with r_C as the number of clusters fixed in advance.
- *Assumption 2*: Complete dataset.
  We assume that there are no missing values for the predictive variables

in the dataset. However, the cluster variable is latent, therefore, its values are always unknown.

- *Assumption 3*: Dirichlet *priors*.
  The parameters of the selective naive Bayes models are assumed to follow a Dirichlet distribution. Thus, α_{ijk} is the Dirichlet hyperparameter for parameter θ_{ijk} from the network, and α_{C-j} is the hyperparameter for θ_{C-j}. Moreover, we have to take into consideration each possible selective naive Bayes whose parameters can be θ_{ijk} or θ_{i-k}. Hence, we assume the existence of both sets of hyperparameters α_{ijk} and α_{i-k}.
- *Assumption 4*: Parameter independence.
  For any possible structure S, the probability distributions $\boldsymbol{\theta}_{ij}$ are random variables which are considered independent for any i and j. Thus, the probability of having the set of parameters $\boldsymbol{\theta}$ for a given structure S can be factorized as follows:

$$p(\boldsymbol{\theta}|S) = p(\boldsymbol{\theta}_C) \prod_{i=1}^{n} \prod_{j=1}^{q_i} p(\boldsymbol{\theta}_{ij}|S) \tag{7.6}$$

- *Assumption 5*: Structure modularity.
  For any possible selective naive Bayes structure, S, we have a prior probability, $p(S)$. The structure modularity assumption states that the prior over structures, $p(S)$, can be decomposed in terms of each variable and its parents:

$$p(S) \propto p_S(C) \prod_{i=1}^{n} p_S(X_i, Pa_i) \tag{7.7}$$

  where $p_S(X_i, Pa_i)$ is the information contributed by variable X_i to the prior over structure S, $p(S)$, and $p_S(C)$ is the information contributed by the cluster variable.

7.3.1.1 E Step (Expectation)

Intuitively, we can see this step as a *completion* of the values for the cluster variable, which are missing. Actually, this step computes the expected sufficient statistics in the dataset given the current parameters of the model, $\breve{\boldsymbol{\theta}}^{(t)}$. These expected sufficient statistics are used in the next step of the algorithm, MA, as if they were actual sufficient statistics from a complete dataset. From now on, $D^{(t)}$ denotes the dataset after the E step at the t-th iteration of the algorithm. Note that although we denote as $\breve{\mathcal{B}}^{(t)}$ the naive Bayes model (structure and parameters) at the t-th iteration of the algorithm, only the parameters $\breve{\boldsymbol{\theta}}^{(t)}$ vary from one iteration to another, being the structure S_{nB} constant throughout the iterations.

The expected sufficient statistics which are used in the MA step to estimate a new model can be obtained as follows:

$$E(N_{ijk}|\breve{\mathcal{B}}^{(t)}) = \sum_{d=1}^{N} p(c^j, x_i^k|\boldsymbol{x}^{(d)}, \breve{\mathcal{B}}^{(t)}) \tag{7.8}$$

where c^j is the j-th value for the class variable, x_i^k is the k-th value for variable X_i, and $\breve{\mathcal{B}}^{(t)}$ is the model that approximates the Bayesian model averaging, that is, a naive Bayes model. The expected sufficient statistic $E(N_{ijk}|\breve{\mathcal{B}}^{(t)})$ denotes, at iteration t, the expected number of cases in the dataset D, where variable X_i takes the value x_i^k, and C takes c^j.

Similarly, we can obtain the expected sufficient statistics for the variable X_i in those selective naive Bayes models where X_i is independent of C and for the cluster variable:

$$\begin{aligned} E(N_{i-k}|\breve{\mathcal{B}}^{(t)}) &= \sum_{d=1}^{N} p(x_i^k|\boldsymbol{x}^{(d)}, \breve{\mathcal{B}}^{(t)}) \\ E(N_{C-j}|\breve{\mathcal{B}}^{(t)}) &= \sum_{d=1}^{N} p(c^j|\boldsymbol{x}^{(d)}, \breve{\mathcal{B}}^{(t)}) \end{aligned} \tag{7.9}$$

Note that, in fact, $E(N_{i-k}|\breve{\mathcal{B}}^{(t)})$ does not depend on the value of C. It denotes the number of cases where the variable X_i takes its k-th value. Therefore, these values are constant throughout the iterations of the algorithm and it is necessary to calculate them only once.

7.3.1.2 MA Step (Model Averaging)

In the classical EM algorithm, the second step is called M (Maximization). In this step the algorithm re-estimates the parameters of the model. Hence, the new parameters approximate the ML or MAP parameter configuration, given the expected sufficient statistics calculated in the previous E step. Instead, the EMA algorithm performs the MA step which obtains a unique naive Bayes model with parameters $\breve{\boldsymbol{\theta}}^{(t+1)}$. These parameters are obtained by calculating $p(c, \boldsymbol{x}|D^{(t)})$ as an average over the MAP configurations for the 2^n selective naive Bayes structures.

In order to make the calculations clearer, we first show how we can obtain $p(c, \boldsymbol{x}|S, D^{(t)})$ for a fixed structure S:

$$p(c, \boldsymbol{x}|S, D^{(t)}) = \int p(c, \boldsymbol{x}|S, \boldsymbol{\theta}) p(\boldsymbol{\theta}|S, D^{(t)}) d\boldsymbol{\theta} \tag{7.10}$$

The exact computation of the integral in Equation 7.10 is intractable, therefore, an approximation is needed (Heckerman, 1995). However, assuming parameter independence and Dirichlet *priors*, and given that the expected sufficient statistics calculated in the previous E step can be used as an approximation to the actual sufficient statistics in the complete dataset, we can approximate $p(c, \boldsymbol{x}|S, D^{(t)})$ by the MAP parameter configuration. This is the parameter configuration that maximizes $p(\boldsymbol{\theta}|S, D^{(t)})$ and can be described in terms of $E(N_{ijk}|\breve{\mathcal{B}}^{(t)})$, and α_{ijk} (Heckerman, 1995; Cooper and Herskovits, 1992). Using the previous considerations, Equation 7.10 results:

$$p(c, \boldsymbol{x}|S, D^{(t)}) \approx \frac{\alpha_{C-j} + E(N_{C-j}|\breve{\mathcal{B}}^{(t)})}{\alpha_C + E(N_C|\breve{\mathcal{B}}^{(t)})} \cdot \prod_{i=1}^{n} \frac{\alpha_{ijk} + E(N_{ijk}|\breve{\mathcal{B}}^{(t)})}{\alpha_{ij} + E(N_{ij}|\breve{\mathcal{B}}^{(t)})}$$

$$= \tilde{\theta}^S_{C-j} \prod_{i=1}^{n} \tilde{\theta}^S_{ijk}$$

where $\tilde{\theta}^S_{ijk}$ is the MAP parameter configuration for S, $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$, $E(N_{ij}|\breve{\mathcal{B}}^{(t)}) = \sum_{k=1}^{r_i} E(N_{ijk}|\breve{\mathcal{B}}^{(t)})$ and similarly for the values related to C.

Note that S is a specific selective naive Bayes structure that represents the dependence between variables. If X_i is independent of C, in Equation 7.11 we should use $E(N_{i-k}|\breve{\mathcal{B}}^{(t)})$ instead of $E(N_{ijk}|\breve{\mathcal{B}}^{(t)})$.

Considering that the structure is not fixed *a priori*, we should average over all selective naive Bayes models in the following way:

$$p(c, \boldsymbol{x}|D^{(t)}) = \sum_S \int p(c, \boldsymbol{x}|S, \boldsymbol{\theta}) \;\; p(\boldsymbol{\theta}|S, D^{(t)}) d\boldsymbol{\theta} \;\; p(S|D^{(t)}) \tag{7.11}$$

Therefore, the model averaging calculations require a summation over 2^n terms, which are the 2^n selective naive Bayes structures with n predictive variables.

Using the previous calculations for a fixed structure, Equation 7.11 can be written as:

$$p(c, \boldsymbol{x}|D^{(t)}) \approx \sum_S \tilde{\theta}^S_{C-j} \prod_{i=1}^{n} \tilde{\theta}^S_{ijk} \;\; p(S|D^{(t)})$$

$$\propto \sum_S \tilde{\theta}^S_{C-j} \prod_{i=1}^{n} \tilde{\theta}^S_{ijk} \;\; p(D^{(t)}|S) \;\; p(S) \tag{7.12}$$

Given the assumption of Dirichlet *priors* and parameter independence, we can approximate $p(D^{(t)}|S)$ efficiently. In order to do so, we adapt the formula to calculate the marginal likelihood with complete data (Cooper and Herskovits, 1992; Heckerman *et al.*, 1995) to our problem with missing values and naive Bayes model. Thus, we have an approximation to $p(D^{(t)}|S)$:

$$p(D^{(t)}|S) \approx \frac{\Gamma(\alpha_C)}{\Gamma(\alpha_C + E(N_C|\breve{\mathcal{B}}^{(t)}))} \prod_{j=1}^{r_C} \frac{\Gamma(\alpha_{C-j} + E(N_{C-j}|\breve{\mathcal{B}}^{(t)}))}{\Gamma(\alpha_{C-j})} \cdot$$

$$\prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + E(N_{ij}|\breve{\mathcal{B}}^{(t)}))} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + E(N_{ijk}|\breve{\mathcal{B}}^{(t)}))}{\Gamma(\alpha_{ijk})} \quad (7.13)$$

At this point, given the structure modularity assumption, we are able to approximate $p(c, \boldsymbol{x}|D^{(t)})$ with the following expression:

$$p(c, \boldsymbol{x}|D^{(t)}) \approx \kappa \sum_{S} \rho^{S}_{C-j} \prod_{i=1}^{n} \rho^{S}_{ijk} \quad (7.14)$$

where κ is a constant and ρ^{S}_{C-j} and ρ^{S}_{ijk} are defined in Equations 7.15 and 7.16 respectively:

$$\rho^{S}_{C-j} = \tilde{\theta}^{S}_{C-j} \;\; p_S(C) \frac{\Gamma(\alpha_C)}{\Gamma(\alpha_C + E(N_C|\breve{\mathcal{B}}^{(t)}))} \cdot \prod_{j=1}^{r_C} \frac{\Gamma(\alpha_{C-j} + E(N_{C-j}|\breve{\mathcal{B}}^{(t)}))}{\Gamma(\alpha_{C-j})} \quad (7.15)$$

$$\rho^{S}_{ijk} = \tilde{\theta}^{S}_{ijk} \;\; p_S(X_i, Pa_i) \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + E(N_{ij}|\breve{\mathcal{B}}^{(t)}))} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + E(N_{ijk}|\breve{\mathcal{B}}^{(t)}))}{\Gamma(\alpha_{ijk})} \quad (7.16)$$

Since we are assuming parameter independence and structure modularity, the calculations for ρ^{S}_{ijk} only depend on X_i and Pa_i. Therefore, if two different structures S_1 and S_2 represent the same relationship between X_i and C, $\rho^{S_1}_{ijk}$ will be the same as $\rho^{S_2}_{ijk}$. Hence, for all the selective naive Bayes models, we only need to calculate ρ_{ijk} (if X_i is dependent on C) and ρ_{i-k} (if X_i is independent of C). The value ρ_{ijk} is calculated as shown in Equation 7.16 and ρ_{i-k} is calculated as ρ_{ijk}, but using $E(N_{i-k}|\breve{\mathcal{B}}^{(t)})$ and α_{i-k}. Thus, Equation 7.14 can be written in terms of ρ_{i-k} and ρ_{ijk} as follows:

$$\begin{aligned} p(c, \boldsymbol{x}|D^{(t)}) \approx \kappa \; \big(& \rho_{C-j} \; \rho_{1-k} \; \rho_{2-k} \; \dots \; \rho_{n-k} \\ + \; & \rho_{C-j} \; \rho_{1jk} \; \rho_{2-k} \; \dots \; \rho_{n-k} \\ \vdots \; & \qquad\qquad \vdots \\ + \; & \rho_{C-j} \; \rho_{1jk} \; \rho_{2jk} \; \dots \; \rho_{njk} \big) \end{aligned} \quad (7.17)$$

We define the symbol Σ_l^{jk} to denote the sum of the product in Equation 7.17 up to the l-th variable.

$$\begin{aligned}\Sigma_l^{jk} &= \rho_{C-j}\ \rho_{1-k}\ \rho_{2-k}\ \cdots\ \rho_{l-k} \\ &+ \rho_{C-j}\ \rho_{1jk}\ \rho_{2-k}\ \cdots\ \rho_{l-k} \\ &\vdots \qquad\qquad \vdots \\ &+ \rho_{C-j}\ \rho_{1jk}\ \rho_{2jk}\ \cdots\ \rho_{ljk}\end{aligned} \tag{7.18}$$

Thus, Σ_n^{jk} can be written in terms of Σ_{n-1}^{jk}

$$\Sigma_n^{jk} = \Sigma_{n-1}^{jk}(\rho_{n-k} + \rho_{njk}) \tag{7.19}$$

Therefore, using the recursive formula in Equation 7.19, we can obtain $p(c, \boldsymbol{x}|D^{(t)})$ as follows (Friedman and Koller, 2003; Dash and Cooper, 2004):

$$p(c, \boldsymbol{x}|D^{(t)}) \approx \kappa\ \rho_{C-j} \prod_{i=1}^{n} (\rho_{i-k} + \rho_{ijk}) \tag{7.20}$$

Note that after the transformations described above and once ρ_{ijk}, ρ_{i-k} and ρ_{C-j} terms have been calculated, the expression $p(c, \boldsymbol{x}|D^{(t)})$ which required a $O(2^n)$ time, can now be evaluated in $O(n)$ time.

In order to calculate the ρ_{ijk}, ρ_{i-k} and ρ_{C-j} terms, we need to set Dirichlet *priors* α_{C-j}, $\alpha_{i-k}, \alpha_{ijk}$ and *priors* over structure $p_S(C)$, $p_S(X_i, Pa_i)$ for all S and $i = 1, \ldots, n$. The values for all these *priors* are assumed to be known. Furthermore, we need the expected sufficient statistics $E(N_{C-j}|\breve{\mathcal{B}}^{(t)})$ and $E(N_{ijk}|\breve{\mathcal{B}}^{(t)})$ which have been calculated in the previous E step in $O(r_C \cdot n \cdot N)$ time.

Now, taking into account the factorization of the joint probability for a naive Bayes model:

$$p(c, \boldsymbol{x}|\boldsymbol{\theta}, S_{nB}) = \theta_C \prod_{i=1}^{n} \theta_{ijk} \tag{7.21}$$

we can observe the similarity between the factorization for a naive Bayes model (Equation 7.21) and the factorization for the Bayesian model averaging of naive Bayes models given by Equation 7.20. Indeed we can calculate the parameters, $\breve{\boldsymbol{\theta}}^{(t+1)}$, for a unique naive Bayes model which approximates a Bayesian model averaging of selective naive Bayes for clustering as follows:

$$\begin{aligned}\breve{\theta}_{ijk}^{(t+1)} &\propto (\rho_{i-k} + \rho_{ijk}) \\ \breve{\theta}_{C-j}^{(t+1)} &\propto \rho_{C-j}\end{aligned} \tag{7.22}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, r_C$.

The parameters for the model that approximates the Bayesian model averaging are calculated in $O(n \cdot N \cdot r_{max} \cdot r_C^2)$ time, where $r_{max} = \max_{1 \leq i \leq n} r_i$. Note that the increment in time complexity in relation to the time needed by the EM algorithm to calculate the ML or MAP parameters, $O(n \cdot N \cdot r_C)$, is insignificant.

Remember that the EMA algorithm calculates the naive Bayes model for clustering iteratively. Therefore, a new naive Bayes for clustering is estimated at each iteration. Hence, the real time complexity of the algorithm is $O(n \cdot N \cdot r_{max} \cdot r_C^2 \cdot It)$ where It is the number of iterations of the EMA algorithm. The EMA algorithm depends on the random initialization of the parameters for the first naive Bayes model, therefore, the total number of iterations may change each time the EMA algorithm is run.

7.3.2 EMA Algorithm for TAN Models

In this section we extend the EMA algorithm described in the previous section in order to perform model averaging calculations over the class of TAN models.

Similarly to the EMA algorithm for naive Bayes models, the EMA-TAN algorithm obtains a single Bayesian network model for clustering which approximates a Bayesian model averaging over the class of TAN models. This is possible by setting an ancestral order among the predictive variables.

First of all, we must define what we mean by the class of TAN models.

Definition 12. *Class of TAN models ($\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$): given an ancestral order $\boldsymbol{\pi}$, a model $\mathcal{B}$ belongs to $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ if each predictive variable has up to two parents (the cluster variable and another predictive variable) and the arcs between variables that are defined in the structure S are directed down levels in the ancestral order $\boldsymbol{\pi}$: $X_j \rightarrow X_i \in S \Rightarrow level_{\boldsymbol{\pi}}(X_i) < level_{\boldsymbol{\pi}}(X_j)$.*

Note that, the classical conception of TAN model allows the predictive variables to form a tree and then, the cluster variable is set as a parent of each predictive variable. However, we do not restrict $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ to only these tree models, but we also allow the predictive variables to form a forest and also the class variable may or may not be set as a parent of each predictive variable. Therefore $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ also includes, among others, naive Bayes and selective naive Bayes models.

For a given ordering $\boldsymbol{\pi}$ and a particular variable X_i, we can enumerate all the possible parent sets for X_i in the class of TAN models $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$. In order to clarify calculations, we superscript with v any quantity related to a predictive variable X_i and thus, we are able to identify the parent set of variable X_i that we are taking into consideration. For example, for a given order $\boldsymbol{\pi} = \langle X_1, X_2, X_3 \rangle$ the possible sets of parents for X_3 in $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ are: $Pa_3^1 = \{\emptyset\}, Pa_3^2 = \{X_1\}, Pa_3^3 = \{X_2\}, \boldsymbol{Pa}_3^4 = \{C, X_1\}, \boldsymbol{Pa}_3^5 = \{C, X_2\}, Pa_3^6 = \{C\}$

with, in this case, $v = 1, \ldots, 6$. In general, we consider, without loss of generality, $\boldsymbol{\pi} = \langle X_1, \ldots, X_n \rangle$ and therefore, for a variable X_i, $v = 1, \ldots, 2i$. Moreover, we use i to index any quantity related to the i-th predictive variable, with $i = 1, \ldots, n$.

Additionally, as well as in the EMA algorithm for naive Bayes models, we need to make the following five assumptions to perform an efficient approximation of model averaging over $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$:

- *Assumption 1*: Multinomial variables.
  Each variable X_i is discrete and can take r_i states. The cluster variable is also discrete and can take r_C possible states, r_C being the number of clusters fixed in advance.
- *Assumption 2*: Complete dataset.
  We assume that there are no missing values for the predictive variables in the dataset. However, the cluster variable is latent; therefore, its values are always missing.
- *Assumption 3*: Dirichlet *priors*.
  The parameters of every model are assumed to follow a Dirichlet distribution. Thus, α_{ijk} is the Dirichlet hyperparameter for parameter θ_{ijk} from the network, and α_{C-j} is the hyperparameter for θ_{C-j}. In fact, as we have to take into consideration each possible model in $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$, the parameters of the models can be denoted as θ_{ijk}^{v}. Hence, we assume the existence of hyperparameters α_{ijk}^{v}.
- *Assumption 4*: Parameter independence.
  The probability of having the set of parameters $\boldsymbol{\theta}$ for a given structure S can be factorized as in Equation 7.6.
- *Assumption 5*: Structure modularity.
  The prior probability $p(S)$ can be decomposed in terms of each variable and its parents (see Equation 7.7).

Parameter independence assumes that the prior on parameters θ_{ijk} for a variable X_i depends only on local structures. This is known as parameter modularity (Heckerman *et al.*, 1995). Therefore, we can state that for any two network structures S_1 and S_2, if X_i has the same parent set in both structures, then $p(\theta_{ijk}|S_1) = p(\theta_{ijk}|S_2)$. As a consequence, parameter calculations for a variable X_i will be the same in every model whose structure defines that the variable X_i has the same parent set.

Theorem 5 (Dash and Cooper, 2004). *There exists, for supervised classification problems, a single model* $\breve{\mathcal{B}} = \langle \breve{S}, \breve{\boldsymbol{\theta}} \rangle$ *which defines a joint probability distribution* $p(c, \boldsymbol{x}|\breve{\mathcal{B}})$ *equivalent to the joint probability distribution produced by averaging over all TAN models. This model* $\breve{\mathcal{B}}$ *is a complete Bayesian network where the structure* $\breve{S}$ *defines the relationship between variables in such a way that, for a variable* X_i*, the parent set of* X_i *in the model* $\breve{\mathcal{B}}$ *is* $\breve{\boldsymbol{Pa}}_i = \cup_{v=1}^{2i} \boldsymbol{Pa}_i^v$.

This theorem can be extended to clustering problems by means of the EMA-TAN algorithm. However, the latent cluster variable prevents the exact calculation of the model averaging and therefore the model $\breve{\mathcal{B}}$ obtained by the EMA-TAN algorithm is not an exact model averaging over $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ but an approximation. Thus, the EMA-TAN algorithm provides a powerful tool that allows to learn a single unsupervised Bayesian network model which approximates Bayesian model averaging over $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$.

The EMA-TAN, as well as the EMA algorithm for naive Bayes models, is an iterative process where the two steps of the algorithm are repeated successively until a stopping criterion is met. At the t-th iteration of the algorithm, a set of parameters, $\breve{\boldsymbol{\theta}}^{(t)}$, for the Bayesian network model $\breve{\mathcal{B}}^{(t)}$ is calculated. Note that, although we differentiate between the Bayesian network models among the iterations of the EMA-TAN algorithm, the structure of the model, $\breve{\boldsymbol{S}}$, is constant and only the parameter set changes. Nevertheless, even though the obtained model is a single unsupervised Bayesian network, its parameters are learned taking into account the MAP parameter configuration for every model in $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$. Thus, the resulting unsupervised Bayesian network will incorporate into its parameters information about the (in)dependencies between variables described by the different TAN models.

7.3.2.1 E Step (Expectation)

The E step for the EMA-TAN algorithm is identical to the E step for the EMA algorithm except for the fact that the class of model structures for the averaging is the class of TAN models, $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$. Thus, this step computes the expected sufficient statistics for each variable X_i and every model in $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ given the current model, $\breve{\mathcal{B}}^{(t)}$. These expected sufficient statistics are used in the next step of the algorithm, MA, as if they were actual sufficient statistics from a complete dataset.

Note that, due to parameter modularity, we do not actually need to calculate the expected sufficient statistics for all the models in $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ because some of these models share the same value for the expected sufficient statistics. Hence, it is only necessary to calculate the expected sufficient statistics with different parent sets. They can be obtained as follows:

$$E(N_{ijk}^{v}|\breve{\mathcal{B}}^{(t)}) = \sum_{d=1}^{N} p(x_i^k, \boldsymbol{Pa}_i^v = j|\boldsymbol{x}^{(d)}, \breve{\mathcal{B}}^{(t)}) \tag{7.23}$$

where x_i^k represents the k-th value of the i-th variable. The expected sufficient statistic $E(N_{ijk}^{v}|\breve{\mathcal{B}}^{(t)})$ denotes, at iteration t, the expected number of cases in the dataset D where variable X_i takes its k-th value, and the v-th parent set of X_i takes its j-th configuration.

Similarly, we can obtain the expected sufficient statistics for the cluster variable. This is a special case since for any model in $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ the parent set for

C is the same (the cluster variable does not have any parent). Therefore, we refuse the use of super-index v in those quantities related only to C.

$$E(N_{C-j}|\breve{\mathcal{B}}^{(t)}) = \sum_{d=1}^{N} p(C=j|\boldsymbol{x}^{(d)}, \breve{\mathcal{B}}^{(t)}) \tag{7.24}$$

Note that, some of the expected sufficient statistics $E(N_{ijk}^{v}|\breve{\mathcal{B}}^{(t)})$ do not depend on the value of C. Since we assume that there are no missing values for the predictive variables, these values are constant throughout the iterations of the algorithm and therefore, they are calculated only once.

7.3.2.2 MA Step (Model Averaging)

In this second step, the EMA algorithm performs the model averaging calculations which obtain a single Bayesian network model with parameters $\breve{\boldsymbol{\theta}}^{(t+1)}$. These parameters are obtained by calculating $p(c, \boldsymbol{x}|D^{(t)})$ as an average over the MAP configurations for the models in $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$.

$$\begin{aligned} p(c, \boldsymbol{x}|S, D^{(t)}) &\approx \frac{\alpha_{C-j} + E(N_{C-j}|\breve{\mathcal{B}}^{(t)})}{\alpha_C + E(N_C|\breve{\mathcal{B}}^{(t)})} \prod_{i=1}^{n} \frac{\alpha_{ijk}^{\mu_i} + E(N_{ijk}^{\mu_i}|\breve{\mathcal{B}}^{(t)})}{\alpha_{ij}^{\mu_i} + E(N_{ij}^{\mu_i}|\breve{\mathcal{B}}^{(t)})} \\ &= \tilde{\theta}_{C-j} \prod_{i=1}^{n} \tilde{\theta}_{ijk}^{\mu_i} \end{aligned} \tag{7.25}$$

where $\tilde{\theta}_{ijk}^{\mu_i}$ is the MAP parameter configuration for $\theta_{ijk}^{\mu_i}$ (μ_i denotes the parent index that corresponds to the parent set for X_i described by S), $\alpha_{ij}^{\mu_i} = \sum_{k=1}^{r_i} \alpha_{ijk}^{\mu_i}$, $E(N_{ij}|\breve{\mathcal{B}}^{(t)}) = \sum_{k=1}^{r_i} E(N_{ijk}|\breve{\mathcal{B}}^{(t)})$ and similarly for the values related to C.

Considering that the structure is not fixed *a priori*, we should average over all model structures in $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ in the following way:

$$p(c, \boldsymbol{x}|D^{(t)}) = \sum_{S} \int p(c, \boldsymbol{x}|S, \boldsymbol{\theta}) p(\boldsymbol{\theta}|S, D^{(t)}) d\boldsymbol{\theta} \;\; p(S|D^{(t)}) \tag{7.26}$$

Therefore, the model averaging calculations require a summation over $2^n n!$ terms, which are the models in $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$.

Using the MAP approximation for the averaging over parameters given a fixed structure, Equation 7.26 can be written as:

$$\begin{aligned} p(c, \boldsymbol{x}|D^{(t)}) &\approx \sum_{S} \tilde{\theta}_{C-j} \prod_{i=1}^{n} \tilde{\theta}_{ijk}^{\mu_i} \;\; p(S|D^{(t)}) \\ &\propto \sum_{S} \tilde{\theta}_{C-j} \prod_{i=1}^{n} \tilde{\theta}_{ijk}^{\mu_i} \;\; p(D^{(t)}|S) \;\; p(S) \end{aligned} \tag{7.27}$$

As well as in the EMA algorithm for naive Bayes models, we can approximate $p(D^{(t)}|S)$ efficiently by adapting the formula to calculate the marginal likelihood with complete data (Cooper and Herskovits, 1992; Heckerman *et al.*, 1995) to our problem. Thus, we have an approximation to $p(D^{(t)}|S)$:

$$p(D^{(t)}|S) \approx \frac{\Gamma(\alpha_C)}{\Gamma(\alpha_C + E(N_C|\breve{\mathcal{B}}^{(t)}))} \prod_{j=1}^{r_C} \frac{\Gamma(\alpha_{C-j} + E(N_{C-j}|\breve{\mathcal{B}}^{(t)}))}{\Gamma(\alpha_{C-j})} \cdot$$

$$\prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij}^{\mu_i})}{\Gamma(\alpha_{ij}^{\mu_i} + E(N_{ij}^{\mu_i}|\breve{\mathcal{B}}^{(t)}))} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}^{\mu_i} + E(N_{ijk}^{\mu_i}|\breve{\mathcal{B}}^{(t)}))}{\Gamma(\alpha_{ijk}^{\mu_i})}$$

At this point, given structure modularity assumption, we are able to approximate $p(c, \boldsymbol{x}|D^{(t)})$ with the following expression:

$$p(c, \boldsymbol{x}|D^{(t)}) \approx \kappa \sum_{S} \rho_{C-j} \prod_{i=1}^{n} \rho_{ijk}^{\mu_i} \tag{7.28}$$

where κ is a constant and ρ_{C-j} and $\rho_{ijk}^{\mu_i}$ are defined in Equations 7.29 and 7.30.

$$\rho_{C-j} = \tilde{\theta}_{C-j} \;\; p_S(C) \frac{\Gamma(\alpha_C)}{\Gamma(\alpha_C + E(N_C|\mathcal{B}^{(t)}))} \prod_{j=1}^{r_C} \frac{\Gamma(\alpha_{C-j} + E(N_{C-j}|\mathcal{B}^{(t)}))}{\Gamma(\alpha_{C-j})} \tag{7.29}$$

$$\rho_{ijk}^{\mu_i} = \tilde{\theta}_{ijk}^{\mu_i} \;\; p_S(X_i, \boldsymbol{Pa}_i^{\mu_i}) \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij}^{\mu_i})}{\Gamma(\alpha_{ij}^{\mu_i} + E(N_{ij}^{\mu_i}|\mathcal{B}^{(t)}))} \cdot$$

$$\prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}^{\mu_i} + E(N_{ijk}^{\mu_i}|\mathcal{B}^{(t)}))}{\Gamma(\alpha_{ijk}^{\mu_i})} \tag{7.30}$$

Since we are assuming parameter independence, structure modularity and parameter modularity, we can apply the dynamic programming solution described in Friedman and Koller (2003), and Dash and Cooper (2004). This process is similar to the one described in the MA step of the EMA algorithm for naive Bayes models (see Section 7.3.1). Thus Equation 7.28 can be written as follows:

$$p(c, \boldsymbol{x}|D^{(t)}) \approx \lambda \;\; \rho_{C-j} \prod_{i=1}^{n} \sum_{v=1}^{2i} \rho_{ijk}^{v} \tag{7.31}$$

with λ being a constant.

Note that the time complexity to calculate the averaging over $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ is in the same order complexity as the time needed to learn the MAP parameter configuration for $\breve{\mathcal{B}}$.

We can see the similarity of the above-described Equation 7.31 with the factorization of a Bayesian network model. Indeed the joint probability distribution of the approximated Bayesian model averaging over $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ for clustering is equivalent to a single Bayesian network model. Therefore, the parameters of the model for the next iteration of the algorithm can be calculated as follows:

$$\begin{aligned} \breve{\theta}_{C-j}^{(t+1)} &\propto \rho_{C-j} \\ \breve{\theta}_{ijk}^{(t+1)} &\propto \sum_{v=1}^{2i} \rho_{ijk}^{v} \end{aligned} \tag{7.32}$$

7.4 Multi-start EMA and Multi-start EMA-TAN Algorithms

The EMA algorithm for naive Bayes models and EMA-TAN algorithm are greedy algorithms that are susceptible to be trapped in a local optimum. The results obtained by the algorithm depend on the random initialization of the parameters. Therefore, we propose the use of a multi-start scheme where m different runs of the algorithm with different random initializations are performed. Then, in order to obtain the final model from the multi-start process we propose three different policies:

- *Uniform Averaging*: The resultant model of the multi-start process is obtained by averaging over the m models obtained throughout the independent runs of the EMA or EMA-TAN algorithm. Using a uniform averaging policy, each one of the m models equally contribute to the final averaged model.
- *Averaging*: The resultant model of the multi-start process is obtained by averaging over the m models obtained throughout the independent runs of the EMA or EMA-TAN algorithm. However, the contribution of each one of the m models to the final model is proportional to its likelihood value.
- *Best Choice*: The resultant model of the multi-start process is obtained by selecting the model with the highest likelihood value among the m models in the multi-start process. This is not a pure Bayesian approach to the model averaging process but it is easier to compute and in practice it may work as well as other more complicated techniques.

7.5 Evaluation of EMA Algorithm for Naive Bayes Models

In this section we present experiments to evaluate several aspects of the EMA algorithm for naive Bayes models. First, we aim to test the accuracy of the approximation to Bayesian model averaging performed by the EMA algorithm by comparing the results with another more accurate but more inefficient method. Then, we evaluate the EMA algorithm as a method for soft model selection; although the model obtained by the EMA algorithm is a naive Bayes model, the parameters of this model should capture the (in)dependence relationships which are present in the model that best fit the dataset. Finally, we evaluate the performance of the EMA algorithm in clustering problems with both synthetic and real data.

7.5.1 Testing EMA versus Brute Force

The EMA algorithm is an approximation to Bayesian model averaging of selective naive Bayes because the existence of a latent cluster variable prevents an exact resolution in closed form of both the averaging over parameters and the marginal likelihood. Actually, only approximations are feasible.

The aim of this section is to demonstrate that the approximation to Bayesian model averaging given by the EMA algorithm is comparable to other more expensive and accurate techniques. Therefore, the EMA model is compared to a model obtained by a brute force approach where the averaging over parameters is also approximated by the MAP configuration but the marginal likelihood is calculated by Gibbs sampling (Geman and Geman, 1984). This brute force method learns the 2^n selective naive Bayes models from the dataset and then averages them over weighted by their posterior probabilities in order to obtain the final model. In order to compare both EMA and brute force models, we propose a comparison test based on Monte Carlo techniques (Shereider, 1964; Sobol, 1984).

Both models for unsupervised classification are estimated from the same dataset. This dataset is sampled from a random selective naive Bayes model.

Since it is computationally very expensive to construct an unsupervised model for clustering by means of a brute force approach, the comparison between the EMA and brute force methods has only been performed for unsupervised classification models with ten and twelve predictive dichotomic variables. The cluster variable is also considered to take only two possible values. On the other hand, the datasets used for the experiments contain 300 samples ($N = 300$). These are not very large datasets, but the number of data samples is high enough to learn the models.

In order to learn the unsupervised model for clustering with both the EMA algorithm and the brute force method, it is necessary to set the *priors* over structures and the hyperparameters. Usually, there is no explicit information about them. Therefore, for the experiment, we choose non-informative values

for those parameters: $\alpha_{ijk} = 1$, $\alpha_{i-k} = 1$, $\alpha_{C-j} = 1$ and $p_S(X_i, Pa_i) = 1$, $p_S(C) = 1$ for all i, j, k.

Each one of the 2^n models for the brute force approach is learned by obtaining an approximation for its MAP parameters using the EM algorithm. Since the EM is a greedy algorithm, we use a multi-start EM[1]. The more times we run the EM algorithm, the more reliable the results are, but we have to find an agreement between efficiency and reliability. In our experiments, the multi-start EM runs the EM algorithm thirty times ($m = 30$) to learn each one of the 2^n selective naive Bayes models. Finally, the brute force model is calculated as an average over the 2^n selective naive Bayes models weighted by the posterior probability for the structure of that model, $p(S|D) \propto p(D|S)p(S)$.

The exact calculation for $p(D|S)$, in a problem with missing values, is also intractable (Cooper and Herskovits, 1992). Since we attempt to compute a reference model to be compared with the model obtained by the EMA algorithm, the approximation to $p(D|S)$ must be as accurate as possible. The most accurate approximations, but also the most time-consuming ones, are obtained by Monte Carlo methods. In this experiment, the approximation to $p(D|S)$ is given by the Candidate method (Chickering and Heckerman, 1997) which is an approximation for the marginal likelihood based on Bayes' theorem and Gibbs sampling (Geman and Geman, 1984).

The EMA, like the EM, is a greedy algorithm. Therefore, we also run a multi-start EMA with $m = 30$. Since the EMA algorithm, in contrast to the EM algorithm, does not maximize the log-likelihood score, we decide to maintain a Bayesian methodology and obtain the final model of the multi-start EMA algorithm by the *Averaging* policy described in Section 7.4.

Once the brute force and the EMA models are obtained, we measure how different they are. This measure is given by the well-known Kullback-Leibler divergence, which is denoted by the following formula (Kullback and Leibler, 1951; Cover and Thomas, 2006):

$$D_{KL}(P_{BF}||P_{EMA}) = \sum_{c,\boldsymbol{x}} p_{BF}(c, \boldsymbol{x}) log_2 \frac{p_{BF}(c, \boldsymbol{x})}{p_{EMA}(c, \boldsymbol{x})} \tag{7.33}$$

where P_{BF} is the probability mass function estimated with a brute force approach and P_{EMA} is the one estimated with the EMA algorithm. This divergence indicates how similar P_{EMA} is with respect to P_{BF}.

In order to know if the difference between both models is significant, the probability distribution of D_{KL} is needed. This is simulated by sampling a large number of random naive Bayes models and measuring their Kullback-Leibler divergence with respect to P_{BF}. In our case, we take $10,000$ naive Bayes models with random parameters. We think this is a large enough number of models and it does not require excessive computational time. Thus, we

[1] The EM algorithm is run m times and the best model among the m runs in terms of log-likelihood is selected. This method is equivalent to the *Best Choice* policy described in Section 7.4.

can test if both probability distributions P_{EMA} and P_{BF} are close to each other.

The experiment described above depends on the random initializations for the EM and EMA algorithms. Therefore, ten independent tests (for models with 10 and 12 predictive variables) have been performed. The results of these tests are shown in Figures 7.2 and 7.3, respectively.

The results of the tests show that, in all of them but two, the EMA is closer to the brute force model than 99% of the random generated models (test with a p-value $= 0.01$, $T_{99\%}$). In fact, only in test 10 from Figure 7.2 and test 1 from Figure 7.3, $D_{KL}(P_{BF}||P_{EMA})$ is greater than the test value $T_{99\%}$. However, the $D_{KL}(P_{BF}||P_{EMA})$ value is very close to $T_{99\%}$, and in both tests it is smaller than a test with a $p\text{-}value = 0.05$ and $p\text{-}value = 0.1$, respectively.

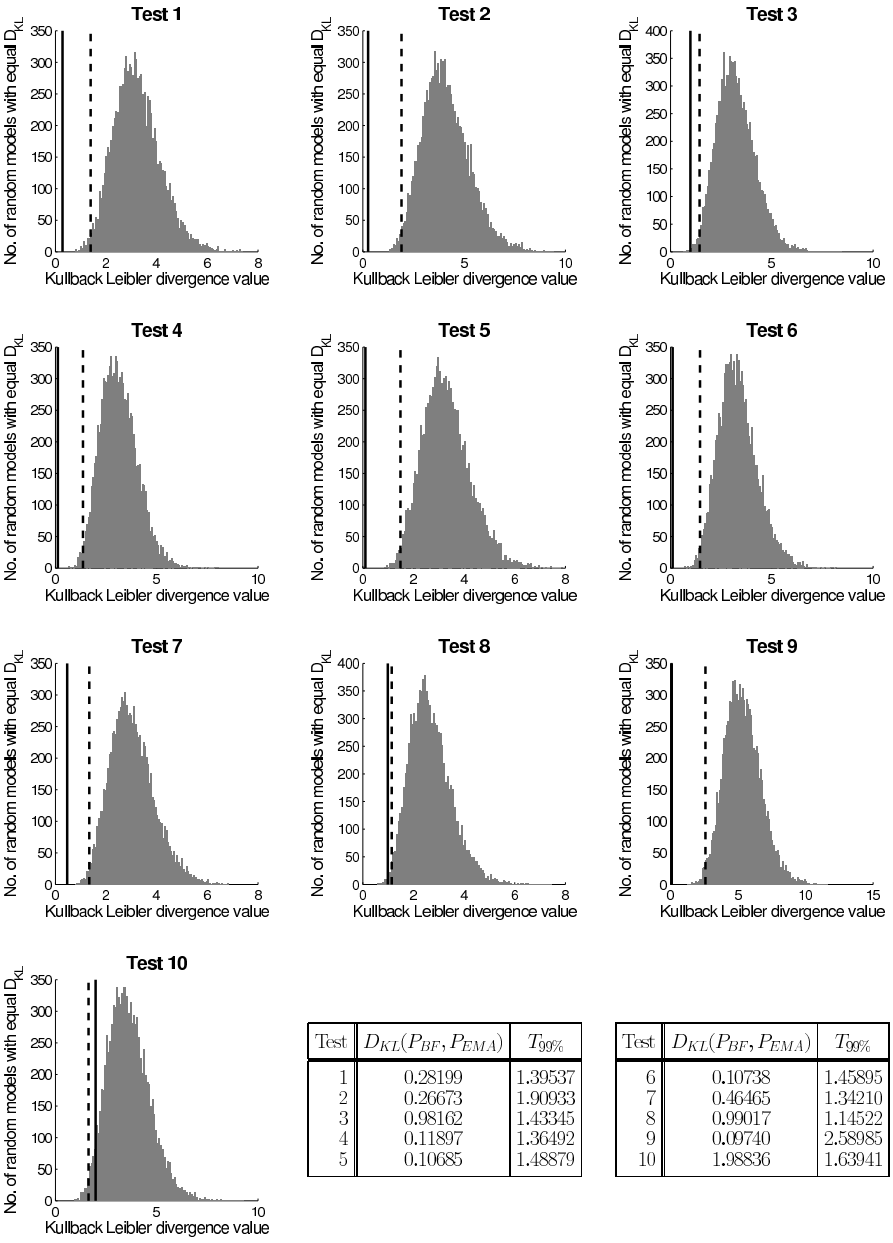
The result of the test for each model can be considered a random variable which follows a binomial distribution $B(10, 0.01)$. In the experiments, for each model, we performed ten independent tests and in, at least, nine of them the EMA model is closer to the brute force model than 99% of the random generated models. The probability of these results is $p(B(10, 0.01) \geq 9) = 10^{-18}$. This is such a small probability that it would be very unlikely to obtain the results shown before if the P_{EMA} and P_{BF} models were not close to each other.

7.5.2 Test for Model Detection

The EMA algorithm is learned by averaging over the MAP parameter configuration for all selective naive Bayes models. Thus, the higher $p(S|D)$ is, the more the model with structure S contributes to the model learned by the EMA algorithm. Therefore, if we sample a dataset, D, from a selective naive Bayes model with structure S, this is supposed to be the structure with the highest $p(S|D)$, that is, the MAP structure. Moreover, as the size of the dataset increases, the peak of the posterior probability function of the structures becomes sharper at the MAP structure. Consequently, as the size of D increases, the MAP model, which is supposed to produce the dataset, makes a higher contribution to the EMA model and thus, the difference between this model and the model used to generate D may decrease.

Following the idea given above, this section shows how a model learned from a dataset by using the EMA algorithm is able to detect the independencies between variables which are described by the model used to generate the dataset. Thus, more empirical evidence about the good performance of the algorithm is provided.

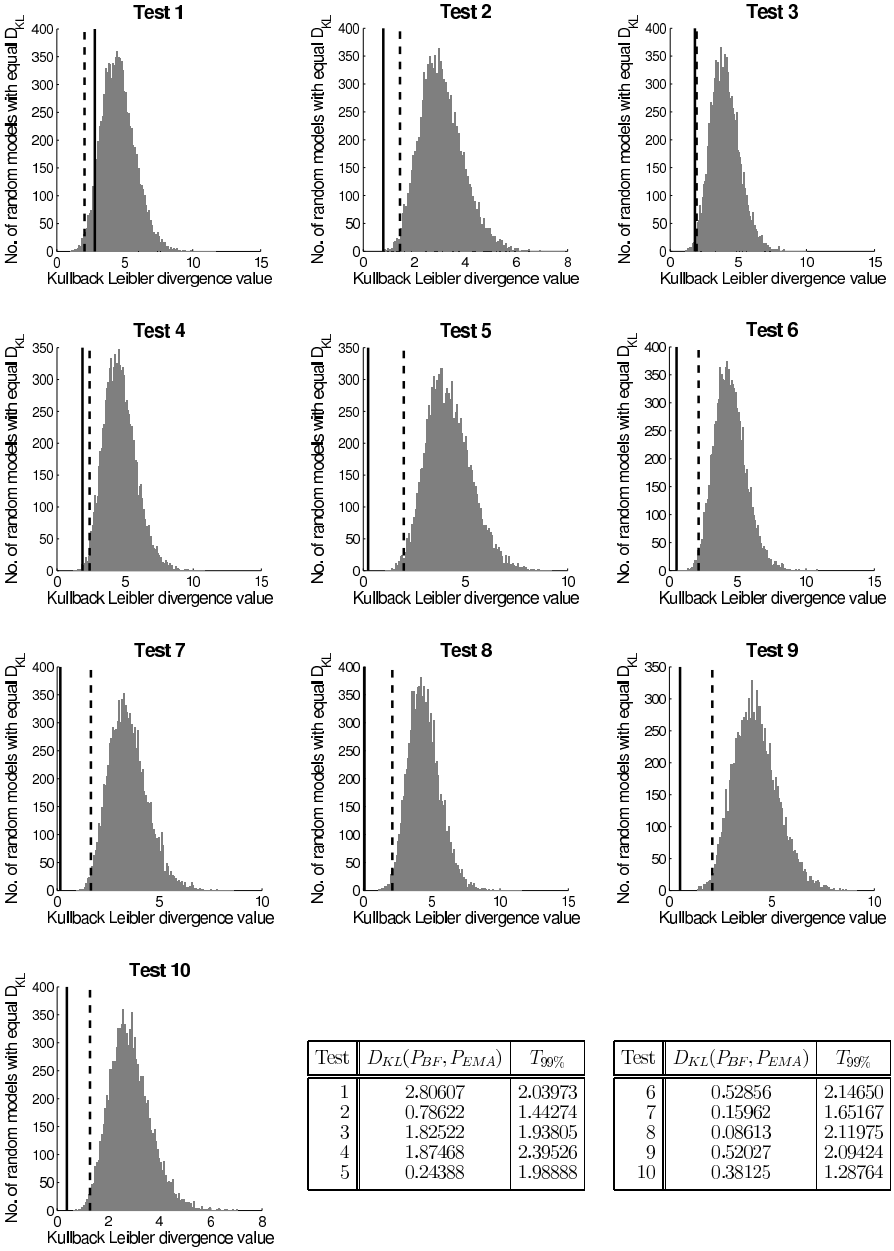
In order to perform the test, a dataset is sampled from a selective naive Bayes model where some of the variables are independent of C. The model learned by means of the EMA algorithm should capture these independencies between predictive variables and C. However, as the independence between variables is not explicitly given in the EMA model, a measure of independence for the variables is needed. This measure of independence is obtained using the



Test	$D_{KL}(P_{BF}, P_{EMA})$	$T_{99\%}$
1	0.28199	1.39537
2	0.26673	1.90933
3	0.98162	1.43345
4	0.11897	1.36492
5	0.10685	1.48879

Test	$D_{KL}(P_{BF}, P_{EMA})$	$T_{99\%}$
6	0.10738	1.45895
7	0.46465	1.34210
8	0.99017	1.14522
9	0.09740	2.58985
10	1.98836	1.63941

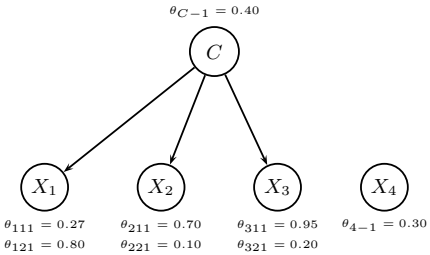
Fig. 7.2. Tests for models with ten predictive variables. The dashed line represents the test with a *p-value* = 0.01, $T_{99\%}$, and the solid line represents the distance between the EMA and brute force models, $D_{KL}(P_{BF}||P_{EMA})$. These same values are given in the tables at the bottom of the figure.



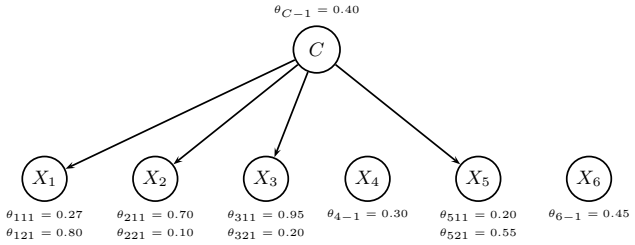
Test	$D_{KL}(P_{BF}, P_{EMA})$	$T_{99\%}$
1	2.80607	2.03973
2	0.78622	1.44274
3	1.82522	1.93805
4	1.87468	2.39526
5	0.24388	1.98888

Test	$D_{KL}(P_{BF}, P_{EMA})$	$T_{99\%}$
6	0.52856	2.14650
7	0.15962	1.65167
8	0.08613	2.11975
9	0.52027	2.09424
10	0.38125	1.28764

Fig. 7.3. Tests for models with twelve predictive variables. The dashed line represents the test with a *p-value* = 0.01, $T_{99\%}$, and the solid line represents the distance between the EMA and brute force models, $D_{KL}(P_{BF}||P_{EMA})$. These same values are given in the tables at the bottom of the figure.

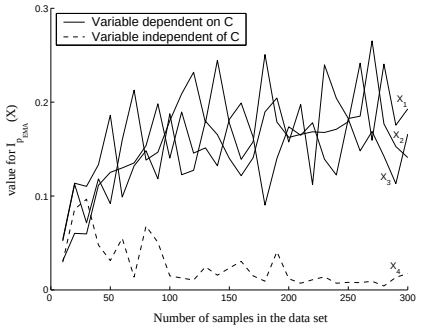


(a) Model with four predictive variables where X_4 is independent of C.

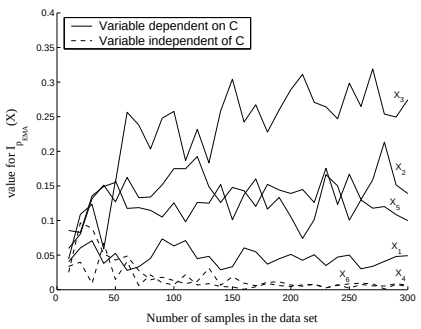


(b) Model with six predictive variables where X_4, and X_6 are independent of C.

Fig. 7.4. Models used in the model detection experiment.



(a) Test with a model with four predictive variables where X_4 is independent of C.



(b) Test with a model with six predictive variables where X_4, and X_6 are independent of C.

Fig. 7.5. Test for model detection.

Kullback-Leibler divergence between $p(X_i)$ and $p(X_i|c^j)$, and it is computed as follows:

$$I_P(X_i) = \frac{\sum_{j=1}^{r_C} D_{KL}(p(X_i)||p(X_i|c^j))}{r_C} \tag{7.34}$$

This measure of independence can be used to rank the predictive variables in terms of how independent of the cluster variable they are. Thus, it is possible to see which predictive variables are more likely to be independent of C. Moreover, comparing the measures of independence for all the predictive variables in the EMA model, the variables which are independent of C in the model used to sample the dataset should obtain smaller values of $I_{P_{EMA}}(X_i)$.

In this test, two different models, which are shown in Figure 7.4, are used to generate the datasets. The parameters for both models have been selected in such a way that the probability distributions of $p(c^0, X_i)$ and $p(c^1, X_i)$, for those X_i which are dependent on C, are not too closed to prevent the EMA algorithm from detecting these dependencies in the dataset. Thus, the models are sampled to generate datasets with different numbers of samples, and an EMA model is learned from each one of the datasets.

The value of $I_{P_{EMA}}(X_i)$ in a model learned by the EMA is sometimes quite noisy due to the random initialization of the parameters. Therefore, the measure of independence is given by the mean of $I_{P_{EMA}}(X_i)$ over a set of models learned via the EMA algorithm. Specifically, we have run the EMA algorithm thirty times in order to obtain thirty different models. Then, $I_{P_{EMA}}(X_i)$ is given by the mean of the measures of independence of X_i over the thirty models.

The results of the tests are shown in Figure 7.5. We can see in this figure that, as the size of the dataset increases, the difference between $I_{P_{EMA}}(X_i)$ for the variables dependent on and independent of C also increases. Consequently, the EMA algorithm is able to detect the independencies between variables revealed in the dataset, and the larger the dataset, the better the EMA algorithm detects the model that generated the data.

7.5.3 Evaluation in Clustering Problems

It is not easy to validate clustering algorithms since clustering problems do not normally provide information about the true grouping of data samples. In general, it is quite common to use synthetic data because the true model that generated the dataset as well as the underlying clustering structure of the data are known. On the other hand, it is possible to use other datasets, such as the ones coming from supervised learning problems, where the true cluster label is also known. This way, the cluster labeling obtained with a cluster algorithm can be compared with the real data partition. In this section, both approaches are taken into consideration in order to evaluate the EMA algorithm in clustering problems.

7.5.3.1 Synthetic Data

In order to illustrate the behavior of the EMA algorithm compared to the classical EM algorithm, we perform an exhaustive test of both algorithms using datasets sampled from randomly generated models, which are obtained by using a modification of the BNGenerator program (Ide *et al.*, 2004).

For a first evaluation of the EMA algorithm in clustering problems, we obtain random selective naive Bayes models where the number of predictive variables vary in $\{4, 6, 8, 10, 20, 40\}$, the number of clusters in $\{2, 3\}$, and each predictive variable can take up to five states. For each selective naive Bayes configuration we generate 50 random models and each one of these models is sampled to obtain different datasets of sizes $10, 20, 40, 80, 160, 320$ and 640. Multi-start EM and multi-start EMA algorithms with $m = 30$ are used to learn the EM and EMA models from each dataset. These models are used to cluster the dataset from which they have been learned. Afterwards, the winner model is determined by comparing the data partition obtained by the EM and EMA models with the true partition of the dataset. As the data partitioning done by clustering methods is sensitive to aliasing (two partitions can be the same but with different cluster labeling), we develop a comparison method insensitive to cluster labeling. This method consists of, for each data partition, obtaining its cluster matrix, that is, a $N \times N$ matrix, A, where a_{ij} with $i = 1, \ldots, N$ and $j = 1, \ldots, N$ is 1 if the i-th and j-th data samples are classified in the same cluster and 0 otherwise. Thus, we can obtain the cluster matrix for the true cluster labels, A_{Real}, and for the cluster labeling obtained by the EMA and EM models, A_{EMA} and A_{EM} respectively. Hence, we can test which partition, EMA or EM, is closer to the real partition by comparing the Hamming distance between A_{Real} and A_{EMA} $-D_H(A_{Real}, A_{EMA})-$, and between A_{Real} and A_{EM} $-D_H(A_{Real}, A_{EM})-$.

In Table 7.1, the results from the experiments with synthetic datasets sampled from random selective naive Bayes models are shown. For each model configuration, the table describes the number of wins/draws/losses of the EMA models with respect to the EM model in relation to the Hamming distances between the models' cluster matrix and A_{Real}. We also provide information about a Wilcoxon signed-rank test used to evaluate whether the Hamming distances $D_H(A_{Real}, A_{EMA})$ and $D_H(A_{Real}, A_{EM})$ are different at the 10% and 1% levels (we write the results in bold if the test is surpassed with a *p-value* < 0.1 and inside a gray color box if the test is suprassed with a *p-value* < 0.01). It can be seen that, in general, the EMA algorithm behaves better than the EM algorithm and the differences between the Hamming distances are, in most of the cases, statistically significant. However, in the largest datasets, the differences between EMA and EM become smaller. In fact, in some simple models, for example the model with 4 variables and 2 clusters, and the model with 6 variables and 2 clusters, the EM algorithm beats the EMA algorithm when the datasets used to learn the models have a high enough number of samples. We hypothesize that this is because, as

the sample size of the dataset increases, the posterior distributions over structures and parameters become sharper, tending to a Kronecker delta function at their MAP configuration. With enough data samples, the EM algorithm is able to approximate this MAP model. In contrast, the EMA algorithm averages over all the possible models and, even when the MAP model contributes the most to the final model, there are other less probable models with smaller contributions that, all together, may add noise to the final model.

#Var	#C	10	20	40	80	160	320	640
4	2	**24/17/9**	**24/18/8**	**19/19/12**	9/25/16	**7/22/22**	**4/22/24**	12/25/13
6	2	**35/6/9**	**34/5/11**	**29/16/5**	**23/19/8**	15/17/18	**11/13/26**	**15/13/22**
8	2	**37/7/6**	**36/6/8**	**26/11/13**	**27/15/8**	**24/14/12**	19/14/17	**13/11/26**
10	2	**33/7/10**	**30/9/11**	**30/10/10**	**28/9/13**	**30/7/13**	22/9/19	20/11/19
20	2	**33/7/10**	**24/17/9**	**25/20/5**	**21/17/12**	**24/23/3**	**22/23/5**	**19/23/8**
40	2	**31/17/2**	**24/22/4**	**23/22/5**	**20/27/3**	**16/31/3**	**15/34/1**	**17/32/1**
4	3	**24/10/16**	22/7/21	**25/6/19**	**26/11/14**	**23/7/20**	**26/9/15**	20/6/24
6	3	**31/3/16**	**34/2/14**	**36/1/13**	**29/2/19**	**33/5/12**	**31/5/14**	26/2/22
8	3	**39/1/10**	**38/0/12**	**40/0/10**	**42/0/8**	**37/0/13**	**33/1/16**	26/2/22
10	3	**32/1/17**	**29/0/21**	**34/1/15**	**35/1/14**	**34/0/16**	**34/3/13**	**29/2/19**
20	3	**31/3/16**	**35/1/14**	**38/1/11**	**40/4/6**	**36/3/11**	**29/3/18**	21/6/23
40	3	**41/1/8**	**44/3/3**	**44/2/4**	**36/5/9**	**34/12/4**	**27/11/12**	**25/13/12**

Table 7.1. Comparison of EMA and EM clustering methods in datasets generated by selective naive Bayes models. Each position of the table indicates the number of wins/draws/losses of the EMA models with respect to the EM models for a specific model configuration and within a dataset size over 50 experiments. Additionally, we write the results in bold if the test is surpassed at the 10% level and inside a gray color box if the test is surpassed at the 1% level.

Note that the experiments described above only use selective naive Bayes models since the EMA algorithm is based on these kinds of models. Naive Bayes and similarly selective naive Bayes are quite simple models and the restrictions that they present are not usually fulfilled in real problems. However, these models have been widely and successfully used in the literature (Cheeseman and Stutz, 1996; Barash and Friedman, 2002; Domingos and Pazzani, 1997; Hand and You, 2001) applied to different problems even if the naive Bayes conditions are not satisfied. Nevertheless, we would like to illustrate the behavior of the EMA algorithm in more realistic problems. Therefore, we repeat the experiment using more complicated models to generate the datasets. In this case, we do not restrict the models used to generate the datasets to the selective naive Bayes family, but we use general Bayesian network classifiers. That is, we randomly generate Bayesian networks varying the number of variables in $\{4, 6, 8, 10, 20, 40\}$, and the maximum number of parents for each variable in $\{2, 3\}$. Each variable can take up to five states. Then, we add the cluster variable, which can take two or three states, and decide randomly which predictive variables are dependent on it.

Table 7.2 shows the results for the experiments with synthetic dataset sampled from random Bayesian network classifiers. It can be seen that, when the Bayesian classifiers used to sample the dataset are quite complex, the

#Var	#C	#Pa	10	20	40	80	160	320	640
4	2	2	**35/11/4**	**25/22/3**	**13/29/8**	13/27/10	12/26/12	18/18/14	**17/19/14**
6	2	2	**31/11/8**	**23/21/6**	14/23/13	**17/23/10**	12/21/17	17/15/18	10/21/19
8	2	2	**30/11/9**	**21/22/7**	12/26/12	14/21/15	15/16/19	**13/16/21**	19/16/15
10	2	2	**27/18/5**	14/19/17	**20/23/7**	**18/19/13**	13/18/19	**18/20/12**	20/15/15
20	2	2	**19/22/9**	14/21/15	19/13/18	**27/11/12**	**24/14/12**	20/8/22	21/9/20
40	2	2	16/19/15	16/15/19	20/17/13	17/13/20	20/5/25	22/10/18	26/7/17
4	2	3	**31/12/7**	**20/24/6**	**18/26/6**	15/24/11	12/20/18	16/22/12	13/26/12
6	2	3	**36/9/5**	**22/17/11**	15/21/14	16/24/10	**18/18/14**	23/18/9	21/12/17
8	2	3	**34/12/4**	**19/21/10**	11/26/13	13/23/14	**17/23/10**	17/16/17	12/13/25
10	2	3	**24/13/13**	18/18/14	**21/20/9**	16/15/19	16/15/19	24/6/20	22/11/17
20	2	3	13/26/11	**24/13/13**	20/16/14	20/12/18	20/10/20	17/11/22	**10/13/27**
40	2	3	17/20/13	21/9/20	18/6/26	23/9/18	20/6/24	23/7/20	18/6/26
4	3	2	**27/7/16**	**30/4/16**	**28/5/17**	**35/1/14**	25/3/23	22/2/26	22/3/26
6	3	2	**36/1/13**	**34/3/13**	**31/1/18**	**30/8/12**	**29/6/15**	**32/5/13**	**27/6/17**
8	3	2	**33/4/13**	**32/0/18**	**39/1/10**	**28/3/19**	24/5/21	**32/0/18**	27/2/21
10	3	2	**30/3/17**	**27/3/20**	29/3/18	27/4/19	27/4/19	28/4/18	26/3/21
20	3	2	**32/0/18**	**35/2/13**	**36/2/12**	**32/2/16**	29/2/19	30/0/20	**32/1/17**
40	3	2	**26/8/16**	23/2/25	**26/2/22**	26/2/22	26/0/24	**32/2/16**	29/1/20
4	3	3	**30/8/12**	**35/3/12**	**29/5/16**	23/8/19	24/3/23	**29/3/18**	23/4/23
6	3	3	**38/3/9**	**35/3/12**	**31/2/17**	17/9/24	24/9/17	**27/8/15**	21/5/24
8	3	3	**30/1/19**	**28/4/18**	**27/5/18**	22/4/24	25/3/22	23/3/24	21/1/28
10	3	3	**35/2/13**	**33/3/14**	**29/3/18**	**32/4/14**	**30/2/18**	30/3/17	**28/0/22**
20	3	3	**31/5/14**	**36/4/10**	**29/1/20**	**31/5/14**	31/2/17	23/2/25	26/1/23
40	3	3	21/7/22	22/7/21	24/1/25	28/0/22	24/2/24	24/0/26	22/0/28

Table 7.2. Comparison of EMA and EM clustering methods in datasets generated by Bayesian network classifiers. Each position of the table indicates the number of wins/draws/losses of the EMA models with respect to the EM models for a specific model configuration and within a dataset size over 50 experiments. Additionally, we write the results in bold if the test is surpassed at the 10% level and inside a gray color box if the test is surpassed at the 1% level.

behavior of both the EMA and the EM algorithms is very similar. This may be because what we learn with both EMA and EM is a naive Bayes model and the dataset contains interrelations between variables which are too complex to be modeled with the restrictions of a naive Bayes. Moreover, the dataset may not have enough samples to capture the complexity of the model that generated the data. On the other hand, except for the most complex models used in the experiments, the EMA algorithm performs better than the EM algorithm. Nevertheless, the differences in the Hamming distances of the cluster matrices for both models are statistically significant in only some experiments.

7.5.3.2 DNA Microarray Data

Nowadays, it is very widespread to use DNA microarrays to monitorize the expression level of thousands of genes at the same time. Although the popularization of different microarray techniques has decreased the experimentation cost, it is still quite expensive. Therefore, microarray datasets normally contain thousands of variables (expression levels of genes) and only a few cases (experiments). Since model averaging techniques account for model uncertainty, they are preferable when only a few data samples are available, which is precisely the case of DNA microarray data. Recently, some model averag-

ing methods have been proposed to deal with both supervised classification (Yeung *et al.*, 2005) and clustering (Medvedovic and Guo, 2005; Vogl *et al.*, 2005) in DNA microarray problems.

The famous AML/ALL-leukemia dataset from the Whitehead Institute (Golub *et al.*, 1999) is one of the first problems that appears in the literature where machine learning techniques are used to classify data from DNA microarrays. The original dataset consists of 72 samples from patients diagnosed with acute lymphoblastic leukemia, ALL (47 samples), and acute myeloid leukemia, AML (25 samples). Each one of these data samples contains the expression value of $7,129$ probes, corresponding to $6,817$ human genes, which were obtained by means of high-density oligonucleotide microarrays produced by Affymetrix.

The use of all the $7,129$ variables to learn a clustering model seems, in principle, nonsense because not all the variables in the original dataset are relevant for clustering purposes and they may blur the real data aggregation. Since the real class label of each data sample is known in this problem, Golub et al. (Golub *et al.*, 1999) propose to filter out variables by selecting only the fifty most informative ones in relation to their correlation with the class variable. This approximation is, in general, unfeasible for clustering problems since the true class label is unknown. By contrast, the EMA algorithm provides a powerful tool that integrates an implicit Bayesian variable selection in the learning process of the clustering model. Thus, although all the variables are included in the clustering model, only the relevant ones are taken into account when clustering the data.

For the experiment, we use all the $7,129$ variables and learn clustering models with both multi-start EM and multi-start EMA algorithms. In the case of the EMA algorithm we use the three different policies of multi-start introduced in Section 7.4.

	Mean	Standard Deviation
multi-start EM	58.86	6.53
multi-start EMA (UA)	79.31	7.13
multi-start EMA (Av)	85.28	10.6
multi-start EMA (BC)	82.50	3.01
SOM	89.47	- -

Table 7.3. Estimated accuracy for clustering methods with Leukemia dataset. The three multi-start EMA policies are used: best choice (BC), uniform averaging (UA) and averaging (Av).

Table 7.3 shows the estimated accuracy for the multi-start EM and multi-start EMA algorithms using, in both cases, $m = 100$. This estimated accuracy value correspond to the percentage of correct classified samples that can be calculated by comparing the cluster labeling obtained by the multi-start

EM and multi-start EMA algorithms with the real cluster partition of the leukemia dataset which, in this specific problem, is known. The table also provides the accuracy value reported in Golub et al. (Golub *et al.*, 1999), where self-organizing maps (SOM) (Kohonen, 2001) are used to cluster the leukemia dataset (the standard deviation value is not reported in the paper). Note that the results obtained by the EMA algorithm can compete with the ones obtained by Golub et al. even when the EMA algorithm uses all the $7,129$ variables, and the SOM only the fifty most informative variables with respect to the class. That is, in this particular case, the EMA algorithm is able to detect the irrelevant variables in the problem and obtain a reasonable estimated accuracy value. In contrast, the multi-start EM algorithm does not have enough data samples to estimate the MAP model and it may consider all the variables equally important for clustering purposes. Thus, the EM algorithm is influenced by irrelevant variables which leads it to obtain a low value for estimated accuracy.

7.6 Evaluation of EMA-TAN Algorithm

In this section we present experiments to evaluate the performance of the EMA-TAN algorithm with synthetic generated data by comparing it with the classical EM algorithm and with the EMA algorithm for naive Bayes models (Section 7.3.1). Similarly to the experiment presented in Section 7.5.3, for EMA-TAN evaluation we obtain random TAN models where the number of predictive variables vary in $\{2, 4, 8, 10, 12, 14\}$, each predictive variable can take up to three states and the number of clusters is set to two. For each TAN configuration we generate 100 random models and each one of these models is sampled to obtain different datasets of sizes $40, 80, 160, 320$ and 640. In the experiments, we compare the multi-start EMA-TAN (called mEMA-TAN for convenience) with three different algorithms:

- *mEM-TAN:* a multi-start EM that learns a TAN model by using, at each step of the EM algorithm, the classical method proposed in Friedman *et al.* (1997) adapted to be used within the EM algorithm.
- *mEM-BNET:* a multi-start EM algorithm used to learn the MAP parameters of a complete Bayesian network for a given order $\boldsymbol{\pi}$.
- *mEMA:* the multi-start model averaging of naive Bayes for clustering.

Note that, for a given order $\boldsymbol{\pi}$, both mEMA-TAN and mEM-BNET models share the same network structure but their parameter sets are calculated in a different way. The number of multi-start iterations for both multi-start EM and multi-start EMA is $m = 100$.

Since the datasets are synthetically generated, we are able to know the real ordering among variables. Nevertheless, we prefer to use a more general approach and assume that this order is unknown. Therefore, we use a random ordering among predictive variables for each mEMA-TAN model that

we learn. As the mEM-BNET algorithm also needs an ancestral order among predictive variables, in the experiments we use the same random ordering for any pair of models (mEMA-TAN vs. mEM-BNET) that we compare.

Every model is used to cluster the dataset from which it has been learned. In the experiments, we compare mEMA-TAN vs. mEM-TAN (Table 7.4), mEMA-TAN vs. mEM-BNET (Table 7.5) and mEMA-TAN vs. mEMA (Table 7.6). For each test, the winner model is obtained by comparing the data partitions obtained by both models with the true partition of the dataset using the method described in Section 7.5.3. Thus, the model with the best estimated accuracy (percentage of correctly classified instances) is the winner model. In Tables 7.4 7.5 and 7.6, the results from the experiments with random TAN models are shown. For each model configuration, the tables describes the number of wins/draws/losses of the mEMA-TAN models with respect to mEM-TAN, mEM-BNET or mEMA models on basis of the estimated accuracy of each model. We also provide information about a Wilcoxon signed-rank test used to evaluate whether the accuracy estimated by two different models is different at the 10% and 1% significance level. We write the results shown in Tables 7.4,7.5 and 7.6 in bold if the test is surpassed at the 10% significance level and inside a gray color box if the test is surpassed at the significance 1% level. It can be seen that, in general, the mEMA-TAN models behave better than the others and the differences between estimated accuracy are, in most of the cases, statistically significant.

We can see that the compared models obtain very similar results when they have a few predictive variables. This is because the set of models that we are averaging over to obtain the mEMA-TAN model is very small. Therefore, it is quite possible that other algorithms such as mEM-TAN select the correct model. Hence, in some experiments with the simplest models (models with 2 predictive variables), mEMA-TAN algorithm significantly lost with the other algorithms. However, when the number of predictive variables in the model increases and the dataset size is relatively big (the smallest datasets are not big enough for a reliable estimation of the parameters) the mEMA-TAN considerably outperforms any other model in the test.

The experimental results from this section reinforce the idea that the results of the Bayesian model averaging outperform other methods when the model that generated the data is included in the set of models that we are averaging over.

7.7 Conclusions and Future Work

In this chapter we have reviewed Bayesian model averaging and the difficulties of using this approach in clustering problems. Additionally, new algorithms are proposed in order to deal with efficient calculations of Bayesian model averaging under restricted class of Bayesian network models such as naive Bayes and TAN. The EMA and EMA-TAN algorithm are efficient in the sense

#Var	40	80	160	320	640
2	11/71/18	**9/71/20**	15/68/17	18/70/12	**15/63/22**
4	37/24/39	**33/17/50**	50/8/42	48/5/47	46/3/51
8	51/6/43	**57/4/39**	**65/6/29**	**74/4/22**	**71/1/28**
10	48/5/47	**59/4/37**	**55/10/35**	**64/6/30**	**67/4/29**
12	**54/4/42**	**65/3/32**	**65/4/31**	**69/4/27**	**83/2/15**
14	**59/8/33**	**68/6/26**	**75/2/23**	**76/4/20**	**76/6/18**

Table 7.4. Comparison between mEMA-TAN and mEM-TAN algorithm in datasets generated by random TAN models. Each position of the table indicates the number of wins/draws/losses of the mEMA-TAN models with respect to the mEM-TAN models for a specific number of predictive variables and within a dataset size over 100 experiments. Additionally, we write the results in bold if the test is surpassed at the 90% level and inside a gray color box if the test is surpassed at the 99% level.

#Var	40	80	160	320	640
2	24/51/25	29/35/36	33/34/33	28/35/37	41/29/30
4	**54/11/35**	51/8/41	**59/8/33**	49/6/45	**57/1/42**
8	**59/8/33**	**64/5/31**	**81/1/18**	**83/0/17**	**91/0/9**
10	**67/7/26**	**76/0/24**	**82/2/16**	**84/0/16**	**94/0/6**
12	**66/5/29**	**86/1/13**	**80/0/20**	**91/0/9**	**89/0/11**
14	**74/2/24**	**83/1/16**	**92/1/7**	**92/0/8**	**96/0/4**

Table 7.5. Comparison between mEMA-TAN and mEM-BNET algorithm in datasets generated by random TAN models. Each position of the table indicates the number of wins/draws/losses of the mEMA-TAN models with respect to the mEM-BNET models for a specific number of predictive variables and within a dataset size over 100 experiments. Additionally, we write the results in bold if the test is surpassed at the 90% level and inside a gray color box if the test is surpassed at the 99% level.

#Var	40	80	160	320	640
2	21/53/26	**24/41/35**	24/48/28	23/45/32	25/45/30
4	47/14/39	49/6/45	49/10/41	50/5/45	56/3/41
8	**52/10/38**	**58/4/38**	**80/4/16**	**90/0/10**	**89/0/11**
10	48/13/39	**55/8/37**	**69/6/25**	**81/1/18**	**88/1/11**
12	**55/8/37**	**78/6/16**	**79/1/20**	**88/2/10**	**88/0/12**
14	**55/11/34**	**68/7/25**	**81/3/16**	**84/3/13**	**92/0/8**

Table 7.6. Comparison between mEMA-TAN and mEMA algorithm in datasets generated by random TAN models. Each position of the table indicates the number of wins/draws/losses of the mEMA-TAN models with respect to the mEMA models for a specific number of predictive variables and within a dataset size over 100 experiments. Additionally, we write the results in bold if the test is surpassed at the 90% level and inside a gray color box if the test is surpassed at the 99% level.

that the approximation to Bayesian model averaging is performed in the same time complexity needed to obtain the ML or MAP parameters by means of the EM algorithm. Moreover, we have shown that the proposed approximation to Bayesian model averaging of naive Bayes models for clustering is equivalent to a single naive Bayes model where the parameters of the model capture the relationship between variables from the selective naive Bayes models used in the averaging. By contrast, the proposed Bayesian model averaging of TAN models for clustering does not result in a TAN model but in a complete Bayesian network model.

With respect to the empirical evaluation of the algorithms, on the one hand, we evaluate the EMA algorithm. We have provided empirical evidence on the fact that the approximation to model averaging obtained by the EMA algorithm is, actually, comparable to model averaging over MAP parameter configurations for selective naive Bayes. Moreover, the EMA algorithm is able to detect the structure of the model that has generated the data. Therefore, the method proposed in this chapter can also be regarded as a Bayesian approach to unsupervised feature subset selection. Additionally, we performed an exhaustive test to illustrate the behavior of the EMA algorithm in clustering problems. We found that the EMA algorithm is a powerful learning algorithm that may be very useful for clustering problems where there are lots of variables (many of them probably irrelevant for clustering purposes) and only a few data samples.

On the other hand, we present an empirical evaluation of the EMA-TAN algorithm. This evaluation is not as exhaustive as the evaluation of the EMA algorithm for naive Bayes: the experiments are based only on synthetic data generated by TAN models. From them, we can conclude that, at least when the model that generated the dataset is included in the set of models that we average over, the averaging over TAN models outperforms other methods that select a single model. However, a more exhaustive evaluation for the EMA-TAN algorithm is interesting for future work in order to know the behavior of the Bayesian model averaging of TAN models for clustering when the dataset is generated from complex models. Nevertheless, the empirical evaluation of the EMA-TAN algorithm is limited by the complexity of the learned model. Note that the resultant model from the EMA-TAN algorithm is a complete Bayesian network model and it is computationally very hard to learn that model for problems with many predictive variables. In order to overcome this situation, an approximation for the Bayesian model averaging of TAN models for clustering can be obtained by restricting the final model to a maximum of k possible parents for each predictive variable. Thus, the calculations will be more efficient making feasible the use of the EMA-TAN algorithm in real problems with a reasonable number of predictive variables.

The EMA-TAN algorithm needs an ordering among the predictive variables. The use of a random ordering is the simplest but, obviously, not the best choice. For instance, Dash and Cooper (2004) adopted a kind of greedy search for selecting a node order for model averaging. Another alternative

is the use of heuristic search methods (Larrañaga *et al.*, 1996). However, a Bayesian approach can be obtained by averaging over several node orders (Hwang and Zhang, 2005). This averaging over several node orders can be extended to clustering problems in order to overcome the dependence on a single node order.

Another major open question is the convergence properties of the EMA and EMA-TAN algorithms. These algorithms attempt to iteratively approximate the Bayesian model averaging model. Hence, the distance between the model learned by the algorithms and the Bayesian model averaging model should decrease at each iteration. For future work, it would also be very interesting to prove the monotonicity of the EMA and EMA-TAN in terms of how this distance decreases throughout the iterations of the algorithms. Additionally, another theoretical point of interest for future studies could be the derivation of an upper bound for the approximation error of the model provided by the EMA and EMA-TAN algorithms in terms of the dataset size and the number of parameters in the model.

Probably one of the limitations for the EMA and EMA-TAN algorithms is that the number of clusters is assumed to be known. Although it is not very usual to know in advance the number of clusters, there are a lot of clustering algorithms (K-means, SOM, EM, etc) with the same limitations. In the literature, we can find several proposals to overcome this problem. For instance, we can learn models with a number of clusters from $r_{C_{min}}$ to $r_{C_{max}}$ and select the best model according to a validity index (Milligan and Cooper, 1985). Furthermore, there are some other techniques proposed to learn the dimensionality of a hidden variable in a Bayesian network classifier. For instance, Elidan and Friedman (2001) proposes a method based on the EM algorithm where the model starts with a *maximal* number of clusters, which are merged in a greedy manner to obtain the final model. Additionally, Karciauskas *et al.* (2004) proposes another method also based on the EM algorithm, where it is not only possible to merge two states of the cluster variable but also to split a cluster into two new different states. It may be very interesting to use these methods in conjunction with the EMA or EMA-TAN algorithm to allow an automatic determination of the number of clusters.

Finally, the expectation model averaging method can be extended to average over more complex model structures in order to capture more complex relationships in the data.

Part IV

Application in Population Genetics

8

Inference of Population Structure Using Genetic Markers

DNA polymorphisms are variations in DNA sequence along individuals within species. Polymorphisms between individuals can arise through several mechanisms which include single nucleotide changes (SNPs), deletions and insertions of nucleotides and variable numbers of short nucleotide sequence repeats (microsatellites). All these polymorphisms occurred during the history of species and are inherited among generations.

Worldwide human population is usually defined in terms of subjective aspects such as language, culture, physical appearance or geographic location. However, human populations also tend to be genetically distant. Genetic differences are caused by a fairly independent evolution under population genetic forces, such as mutation, recombination, random drift and selection. This variation within and between populations can be observed at genetic marker locations. Recent studies using a variety of genetic markers, have shown that individuals sampled worldwide fall into clusters that roughly correspond to continental lines as well as to self-identifying racial groups (Rosenberg *et al.*, 2002; Bamshad *et al.*, 2003; Rosenberg *et al.*, 2005; Corander and Marttinen, 2006).

The information about population structure, namely population stratification and admixture, is useful not only in evolutionary studies or subspecies classification (Pritchard *et al.*, 2000b; Rosenberg *et al.*, 2002) but also in association studies of disease genes (Sillanpää *et al.*, 2001; Patterson *et al.*, 2004; Riddle *et al.*, 2006; Price1 *et al.*, 2006). Association studies often use a case-control design to identify genetic variants related to a specific disease by comparing allele frequencies between unrelated individuals that are affected and those unaffected. However, the presence of population stratification can lead to spurious allelic association between candidate marker and a phenotype (Pritchard *et al.*, 2000a; Cardon and Palmer, 2003).

From a machine learning point of view, the inference of population structure can be seen as a clustering process where individuals are assigned to their population of origin according to their DNA polymorphisms. In the literature, two main clustering approaches to the inference of population structures can

be found: distance-based methods and model-based methods. Distance-based methods use pairwise distances between individuals to obtain a clustering partition of the population (Bowcock *et al.*, 1994). These methods are highly dependent on the selected distance measure and therefore it is very difficult to know if the obtained clustering partition is meaningful. On the other hand, model-based clustering assumes that there is a generative probabilistic model underlying the genetic information of the individuals. Another key modeling assumption is linkage and Hardy-Weinberg disequilibrium. Since these models are based on probability theory, a large amount of methods from statistical learning, sampling theory and Bayesian statistics can be used. Bayesian statistical methods based on Markov chain Monte Carlo (MCMC) are commonly used for the inference of population structure. Particularly, STRUCTURE (Pritchard *et al.*, 2000b; Falush *et al.*, 2003) is one of the most widely used algorithms based on MCMC. However, there are other proposals such as PARTITION (Dawson and Belkhir, 2001), BAPS 2 (Corander *et al.*, 2004; Corander and Marttinen, 2006), a spatial statistical model for landscape genetics proposed by Guillot et al. (2004), or the learning of mixtures of trees (Kollin and Koivisto, 2006). Additionally, there are other algorithms, for instance methods based on the EM algorithm such as PSMIX (Wu *et al.*, 2006), or based on information theory (O'Rourke *et al.*, 2005).

Even when high-throughput technologies offer the possibility of measuring a large number of polymorphisms simultaneously, it has sometimes been observed, for certain ancestry inference procedures, that accuracy of inference does not necessarily increase as markers are accumulated. Indeed, in an increasing number of species, the number of markers from which allele frequencies are available exceeds those required for accurate assignments. That is, it is possible to find robust clustering patterns by using a panel with only a part of the markers which are available (Turakulov and Easteal, 2003; Rosenberg *et al.*, 2003; Rosenberg, 2005). Thus, not only may the accuracy and efficiency of the population inference method be improved but also the genotyping cost reduced. Nevertheless, there is not a clear criteria to select the set of markers needed to obtain a robust clustering partition. Furthermore, the fact that not all the markers available are needed, suggests that there is redundant information and/or markers that are not relevant to cluster the individuals into their population of origin. This redundant and irrelevant information may damage the ability of the clustering methods to infer the population structure.

In this chapter, we adapt the multi-start Expectation Model Averaging (multi-start EMA) algorithm from Section 7.3.1 to infer the structure of a population. The Bayesian model averaging process performed by the EMA algorithm incorporates a kind of implicit feature selection in the model. Therefore, we can use the information contained in this model to filter out those genetic markers which are considered irrelevant for obtaining the population structure. The information about relevant markers may be useful in many other genetic studies. Hence, we propose a two-step test based on mutual in-

formation that can be used to retrieve this information from the clustering model.

8.1 Notation in the Problem Context

The marker loci are denoted by $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ and the cluster variable, C, represents the population grouping of the N polyploid individuals $D = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$. Since we would like to use the proposed method with human genetic data, from now on we assume diploid data. Therefore the d-th individual of the population is characterized by n genetic markers, $\boldsymbol{x}^{(d)} = \left(x_1^{(d)}, \ldots, x_n^{(d)}\right)$ and each marker X_i, with $i = 1, \ldots, n$, contains information for two alleles $x_i^{(d)} = \{x_i^{(d),h_1}, x_i^{(d),h_2}\}$. Capital letters, X_i, represent a marker locus and small letters, x_i, denote a specific allele value for the marker locus.

The model underlying the population structure is assumed to be a naive Bayes. However, the learning process performed by the multi-start EMA includes a kind of implicit selection of relevant marker loci in the model. We say that it is an implicit selection because all the markers are included in the naive Bayes model learned by the multi-start EMA, but the learning process gives rise to the fact that not all the markers have the same significance when obtaining the population structure.

8.2 Modification of the EMA algorithm

The aim of the algorithm is to obtain a clustering model that provides a posterior distribution over the dataset and thus, allows to infer the population structure of the individuals under study. For this purpose, we adapt the EMA algorithm introduced in Section 7.3.1 to deal with polyploid [1] and missing data. Thus, we provide a useful and realistic tool for the inference of population structure.

The parameters of the model $\breve{\boldsymbol{\theta}}^{(t)} = \left(\breve{\theta}_{ijk}^{(t)}, \breve{\theta}_{C-j}^{(t)}\right)$, in the context of the inference of population structure, denote the frequency of every allele k at marker X_i in a population j, $\breve{\theta}_{ijk}^{(t)}$, and the prior probability of each population j, $\breve{\theta}_{C-j}^{(t)}$. In order to calculate these parameters, the algorithm takes into account the possibility that each marker X_i can be relevant or irrelevant for the inference of the population structure. In the calculations, the frequency at marker X_i of the allele k in population j when marker X_i is considered relevant

[1] Although diploid data are assumed in this work, the algorithm is parametrized to be used with any ploidy. The ploidy being the number of homologous sets of chromosomes in a biological cell.

is represented as θ_{ijk} . However, if it is considered irrelevant, we take into consideration the overall frequency of allele k, represented as θ_{i-k}. Additionally, all the assumptions from Section 7.3.1 but the complete dataset assumption are needed to perform the model averaging calculations efficiently. In order to relax the complete dataset assumption and in order to use the EMA algorithm with polyploid data, the E step from the EMA algorithm must be modified. Therefore, the E step is similar to the one described in Section 7.3.1 in the sense that in this step we compute, given the current model parameters $\breve{\boldsymbol{\theta}}^{(t)}$, the expected number of individuals from a population j that present allele k at the i-th marker when the marker is considered relevant for clustering purposes, $E(N_{ijk}|\breve{\boldsymbol{\theta}}^{(t)})$, or irrelevant $E(N_{i-k}|\breve{\boldsymbol{\theta}}^{(t)})$, and the expected number of individuals classified into population j, $E(N_{C-j}|\breve{\boldsymbol{\theta}}^{(t)})$. However, expected sufficient statistics are now computed as follows:

$$
\begin{aligned}
E(N_{ijk}|\breve{\boldsymbol{\theta}}^{(t)}) &= \sum_{d=1}^{N} p(c^j, x_i^{k,h_1}|\boldsymbol{x}^{(d)}, \breve{\boldsymbol{\theta}}^{(t)}) + p(c^j, x_i^{k,h_2}|\boldsymbol{x}^{(d)}, \breve{\boldsymbol{\theta}}^{(t)}) \\
E(N_{i-k}|\breve{\boldsymbol{\theta}}^{(t)}) &= \sum_{d=1}^{N} p(x_i^{k,h_1}|\boldsymbol{x}^{(d)}, \breve{\boldsymbol{\theta}}^{(t)}) + p(x_i^{k,h_2}|\boldsymbol{x}^{(d)}, \breve{\boldsymbol{\theta}}^{(t)}) \\
E(N_{C-j}|\breve{\boldsymbol{\theta}}^{(t)}) &= \sum_{d=1}^{N} p(c^j|\boldsymbol{x}^{(d)}, \breve{\boldsymbol{\theta}}^{(t)})
\end{aligned}
\tag{8.1}
$$

Note that, as the structure of the model is constant throughout the iterations of the algorithm, we neglect the structure in the notation and only the parameters are used. Moreover, we abuse the notation by using c^j and x_i^{k,h_g}, with $g = 1, 2$, to denote the fact that the cluster variable C takes the j-th value and marker X_i takes the k-th value for the genetic copy h_g respectively. Although we avoid the use of superscript d in order to clarify the notation, the information contained in c^j and x_i^{k,h_g} is assumed to belong to the d-th individual. Moreover, for a simpler notation, when we write $\boldsymbol{x}$, we assume that each marker X_i takes its k-th allele value. Note that, both copies of the genetic information for each marker, $x_i = \{x_i^{k,h_1}, x_i^{k,h_2}\}$, are taken into account for the calculation of $E(N_{ijk}|\breve{\boldsymbol{\theta}}^{(t)})$ and $E(N_{i-k}|\breve{\boldsymbol{\theta}}^{(t)})$. In the original EMA algorithm, as missing values were not allowed, the values of $E(N_{i-k}|\breve{\boldsymbol{\theta}}^{(t)})$ were constant throughout the iterations of the algorithm. However, the current modification proposed in this chapter allows the presence of missing data and therefore not only the value of $E(N_{ijk}|\breve{\boldsymbol{\theta}}^{(t)})$ and $E(N_{C-j}|\breve{\boldsymbol{\theta}}^{(t)})$ but also $E(N_{i-k}|\breve{\boldsymbol{\theta}}^{(t)})$ may change at each iteration.

The calculations for the MA step of the EMA algorithm are exactly the same as the ones described in the EMA algorithm from Section 7.3.1. Additionally, for the multi-start EMA algorithm we adopt the best choice policy,

where the best model, in terms of likelihood, among the m models calculated by the multi-start process, is selected to be the final model (see Section 7.4).

8.2.1 Selecting the Most Relevant Markers for Population Inference

The model averaging process performed by the EMA algorithm can also be perceived as an implicit unsupervised feature selection that is incorporated in the final model. In fact, although the EMA algorithm, and consequently the multi-start EMA, obtains a naive Bayes model where all the maker loci are independent given population assignment, the parameters of the resultant model are calculated by a model averaging over selective naive Bayes. Thus, these parameters should reflect the significance of each marker for the inference of population structure.

In this section we propose a two-step test that can be used to obtain information about relevant markers that is implicitly contained in the final naive Bayes model calculated by the multi-start EMA. This test is based on mutual information. It is known (Pardo, 1997; Cover and Thomas, 2006) that the statistic $2NI(X_i, C)$, where $I(X_i, C)$ is the mutual information between X_i and C, asymptotically follows, under the null hypothesis of independence between X_i and C, a Chi-square probability distribution with $(r_i - 1)(r_C - 1)$ degrees of freedom. In our case, r_i is the number of different alleles that a marker X_i can present and r_C the number of clusters.

The mutual information between a marker X_i and the cluster variable, or the mutual information between two markers X_i and $X_{i'}$ with $i \neq i'$ can be calculated using the naive Bayes model obtained by the multi-start EMA. Thus, a Chi-square test can be performed to decide which marker loci are relevant for the clustering process. In the first step, a test threshold p_{rel} is set and a Chi-square test is used to filter out those markers which are considered not relevant for clustering purposes. This first step selects the relevant markers but the set of selected markers may contain redundant information. As a consequence, we develop a second step to filter out redundant information by again using a Chi-square test with a test threshold p_{red}. In this second step the pairwise mutual information of the markers selected in the first step is calculated and it is used to decide whether or not two markers are redundant. Figure 8.1 describes the two-step algorithm to obtain the set of markers, $\boldsymbol{X}_{rel}$, which are relevant to obtain the underlying population structure.

The thresholds p_{rel} and p_{red} can be used to control the number of selected markers. On the one hand, the higher the p_{rel} value is, the more markers are selected as relevant for clustering. On the other hand, as p_{red} decreases, the number of markers considered redundant increases and therefore, the final number of selected markers is smaller.

$\boldsymbol{X}_{rel} = (X_1, \ldots, X_n)$

– STEP 1 –

for $i = 1$ **to** n
  if $2NI(X_i, C) < \chi^2_{(r_i-1)(r_C-1);1-p_{rel}}$
    remove X_i from $\boldsymbol{X}_{rel}$
  end if
end for

– STEP 2 –

for all $X_i, X_{i'}$ **with** $X_i, X_{i'} \in \boldsymbol{X}_{rel}$ **and** $i \neq i'$
  if $2NI(X_i, X_{i'}) < \chi^2_{(r_i-1)(r_{i'}-1);1-p_{red}}$
    if $I(X_i, C) < I(X_{i'}, C)$
      remove X_i from $\boldsymbol{X}_{rel}$
    else
      remove $X_{i'}$ from $\boldsymbol{X}_{rel}$
    end if
  end if
end for

Fig. 8.1. Pseudo-code for Marker Selection Algorithm.

8.2.2 Number of Clusters and Genetic Distance

The multi-start EMA algorithm requires the specification of the number of clusters underlying the population. Since the real number of groups is usually unknown, we propose to investigate the number of subpopulations by evaluating runs of the algorithms with a different number of clusters. The different configurations for the number of clusters can be compared by using the genetic distance F_{ST}. This genetic distance is a measure of the dissimilarity of genetic material between different species or individuals of the same species (Reynolds *et al.*, 1983; Weir, 1996), and it can be interpreted as the proportion of the total genetic variance contained in subpopulations relative to the total genetic variance.

F_{ST} metric is computed as $F_{ST} = -ln\left(1 - \frac{\sum_i a_i}{\sum_i (a_i + b_i)}\right)$.The calculations of a_i and b_i are taken from (Reynolds *et al.*, 1983) and adapted to our specific notation:

$$a_i = \frac{2\left[\sum_j N\theta_{C-j} \sum_k (\theta_{ijk} - \sum_j \theta_{ijk})^2 - b_i(r_C - 1)\right]}{2N(r_C - 1)(1 - \sum_j \theta^2_{C-j})}$$

$$b_i = \frac{2\sum_j \left(N\theta_{C-j}(1 - \sum_k \theta^2_{ijk})\right)}{2N - r_C} \tag{8.2}$$

The F_{ST} distance gives some intuition about how far the analyzed subpopulations are. Certainly, the quantity can range from 0 to 1 and it increases as the sample allele frequencies of individuals from different subpopulations diverges. Since we aim to obtain clusters which represents well differentiated populations, the F_{ST} may be a good metric to evaluate the quality of the clustering partition from a biological point of view. Therefore, in order to decide the number of clusters, we propose to compare the mean F_{ST} metric over a set of ten independent runs among experiments with several numbers of clusters. Additionally, it is possible to perform a Mann-Whitney test to decide if the differences in the F_{ST} metric are statistically significant. Thus, we may be able to decide the proper number of clusters in the dataset.

8.3 First Approach to the Inference of Population Structure: A Toy Example

In order to show how the characteristics of the EMA algorithm can fit properly to the population substructure problem, we firstly use the toy example introduced in (O'Rourke *et al.*, 2005). The dataset for this toy example is composed of individuals represented by 50-character bit-strings. We generate three ancestral sequences with relative Hamming distances 2, 3 and 5. Each ancestral sequence is used to generate a set of 19 new clone individuals in which random mutations may happen at each position of the string with a mutation rate 0.05. Thus, we obtain 60 individuals grouped into three different populations with an expected Hamming distance of 2.5 between a mutant and its ancestral sequence. The individuals from the three different populations are differentiated by 5 positions of the sequence and the rest of the positions may only differ from one individual to another because of random mutations. In fact, it is enough with only 2 of the 5 positions to differentiate between individuals from the three populations because there is redundant information.

We use a multi-start EMA algorithm with $\epsilon = 0.01$ and $m = 1000$ runs in the multi-start process to cluster the 60 individuals into 3 clusters. Since the EMA algorithm is not deterministic, we perform 10 runs of the multi-start algorithm that, on average, classify 95% of the sequences into their original population with null standard deviation. Moreover, the naive Bayes model obtained by the multi-start EMA at each run is used to obtain the set of relevant positions to decide the clustering membership. The test described in Section 8.2.1 with parameters $p_{rel} = 0.01$ and $p_{red} = 0.01$ is used to obtain these relevant positions. All the 10 runs yield the selection of only two positions of the five positions that differentiate between the three populations. Thus, the algorithm correctly selects the minimum number of positions to differentiate between populations.

Moreover, the STRUCTURE software (Pritchard *et al.*, 2000b) with default parameters and $10,000$ models for the burning period and also $10,000$

iterations to learn the model is able to classify, on average over ten runs, only 79.64% of the sequences into their population of origin with standard deviation 9.91%.

The proposed toy example shows that the EMA algorithm presents a good behavior for clustering sequences from different populations and where random mutation across the sequence positions may happen. In the following sections we apply the EMA algorithm to real datasets.

8.4 Inference of Population Structure using Single Nucleotide Polymorphisms

For this experiment, we use the dataset of common SNPs reported in Hinds *et al.* (2005). This dataset contains about 1.5 million SNPs uniformly distributed across the human genome and which are common to, at least, individuals from the three human populations under study: European, African and Asian. The data came from the genotype of 71 unrelated individuals: 24 European-American, 23 African-American and 24 Han-Chinese from the Los Angeles area. This data is publicly available from the NCBI's SNP database (dbSNP) build 123 ascension numbers $ss23145044$ to $ss24731426$.

Following the experimental description from (O'Rourke *et al.*, 2005), and in order to avoid linkage disequilibrium from proximity in the genome, we only use every thousandth SNP, leaving a total of $1,520$ marker loci. Although the EMA algorithm is able to deal with polyploid data, we follow the approach to the problem presented in O'Rourke *et al.* (2005). Therefore, each individual is split in two haploid sequences belonging to the same population. Hence, the dataset is made up of 142 sequences with $1,520$ SNPs each. On the other hand, this dataset with $1,520$ SNPs contains information about SNPs from all over the human genome. However, in other real problems, not all this information is always available. Sometimes only SNPs from one or several chromosomal segments are provided and then, the population substructure is more difficult to retrieve. In order to simulate these situations, we split the dataset into 19 datasets with only 80 SNPs. In the experiments we refer to the dataset that contains all the $1,520$ SNPs as $dsSNPs$. The smaller datasets with 80 SNPs are denoted as $ds1, \ldots, ds19$.

Table 8.1 shows the percentage of individuals, on average over 10 independent runs, correctly assigned to their population of origin using both multi-start EMA and STRUCTURE algorithms for each dataset. The parameters used for both algorithms are the same as those used in the toy example. A Man-Withney test at 0.01 level is performed to check if the difference between the two algorithms is statistically significant. Those values which are significantly better are written in bold in Table 8.1. We can see that the results obtained by the multi-start EMA algorithm outperform the STRUCTURE software, and the differences are statistically significant for all the datasets.

	STRUCTURE		**EMA**	
Dataset	Mean	Std	Mean	Std
$ds1$	87.28	0.55	**95.07**	0.47
$ds2$	88.27	1.35	**93.24**	0.36
$ds3$	88.48	0.48	**93.38**	1.49
$ds4$	91.27	0.34	**95.77**	0.00
$ds5$	83.75	7.67	**93.94**	0.36
$ds6$	84.53	0.51	**86.13**	0.48
$ds7$	92.20	0.25	**96.48**	0.00
$ds8$	74.79	2.66	**82.75**	2.11
$ds9$	81.86	0.62	**86.06**	1.78
$ds10$	85.28	0.74	**89.08**	1.11
$ds11$	87.25	0.38	**90.14**	0.00
$ds12$	89.59	0.35	**91.55**	0.00
$ds13$	90.72	0.38	**94.37**	0.00
$ds14$	89.41	0.26	**95.99**	0.48
$ds15$	86.08	0.30	**90.07**	0.52
$ds16$	84.96	0.84	**93.45**	0.48
$ds17$	88.34	0.17	**93.52**	0.30
$ds18$	81.16	0.53	**86.55**	2.92
$ds19$	82.65	0.86	**86.62**	0.00
Mean over $ds1$-$ds19$	86.20	—	**91.27**	—
$dsSNPs$	96.78	0.09	**100**	0.00

Table 8.1. Percentage of correctly assigned individuals to their population of origin (mean and standard deviation over 10 runs) using both EMA and STRUCTURE algorithms.

The SSCC method proposed in O'Rourke *et al.* (2005) also obtains 100% of correct classications for $dsSNPs$ dataset. In the case of the dataset with only 80 SNPs, only the mean value over the 19 datasets is reported in the paper. The multi-start EMA algorithm obtains, on average over the 19 datasets, 91.27% of correct classified individuals. This figure is very close to the precision obtained by the SSCC, 91.80%

Another important characteristic of the EMA algorithm is its ability to select the set of relevant SNPs for clustering purposes. Using the $dsSNPs$ dataset and setting $p_{rel} = p_{red} = 0.01$, the 10 runs of the multi-start EMA algorithm give rise to 10 sets of selected relevant SNPs. These datasets contain, on average, 277.1 relevant SNPs and 146 of them are common to the 10 sets. Note that each set of relevant SNPs is selected on the basis of a different naive Bayes model learned with the multi-start EMA algorithm. Therefore, although two sets of relevant SNPs share many SNPs, some of them may be different. The fact that two sets of relevant SNPs contain different SNPs does not necessarily mean that one set is better than the other because they may contain different subsets of non-redundant SNPs. In order to evaluate the sets of relevant SNPs obtained by the multi-start EMA algorithm, we run the STRUCTURE software with the same parameter configuration as before on

these datasets. The percentage of correctly classified individuals is, on average over the 10 different sets of relevant SNPs, 94.98% with standard deviation 4.66%. Note that the reduction in the number of SNPs in the dataset is very big while the percentage of correctly classified individuals is similar.

8.5 Inference of Population Structure Using Different DNA Polymorphisms

In this section we aim to evaluate the behavior of the EMA algorithm for the inference of population structure in a more complex problem. The data for this experiment were taken from the HGDP-CEPH human genome diversity cell line panel. The diversity cell line panel is a large and widely-used collection of DNA samples from individuals distributed around the world. The properties of the sample of individuals were first reported by (Cann *et al.*, 2002). However, new genotypes have been reported since then. Specifically, we employ a dataset included in the HGDP-CEPH which has been used in recent studies of genetic structure of human populations (Rosenberg *et al.*, 2005). The dataset contains 993 markers, including 783 microsatellites and 210 insertion/deletion polymorphisms corresponding to 1048 individuals from 53 different human populations distributed around the world. Although the individuals are classified into 53 human populations or ethnic groups, they correspond to seven major regions: Africa, Europe, Middle East, Central/South Asia, East Asia, Oceania and America.

The aim of the study is to group individuals into genetic clusters in such a way that for each individual an estimated membership coefficient for each cluster is given. This probability can be seen as an admixture coefficient since each individual may have genetic information belonging to different population sources.

The proposed algorithm is compared with the well known STRUCTURE software (Pritchard *et al.*, 2000b; Falush *et al.*, 2003) version 2.1. Both the multi-start EMA and STRUCTURE require a prespecified number of clusters, therefore, we run experiments with two, three and four numbers of clusters. Since the obtained clustering results depend on the random initialization of the models, we run ten executions of each algorithm with each number of clusters. For the multi-start EMA we use a $\epsilon = 0.01$ and 500 iterations in the multi-start process, $m = 500$. For STRUCTURE, we use the configuration reported in (Rosenberg *et al.*, 2005) where an allele frequency correlated model is used. Additionally, the parameters are calculated using 1000 iterations after a burn-in period of 5000.

Figure 8.2 shows the estimation of population structure obtained by the multi-start EMA with two, three and four clusters ($r_C = 2$, $r_C = 3$ and $r_C = 4$). Similarly, Figure 8.3 shows the results obtained by STRUCTURE. The plots were generated with DISTRUCT (Rosenberg, 2004), where each individual is represented by a segment partitioned into r_C colored parts that

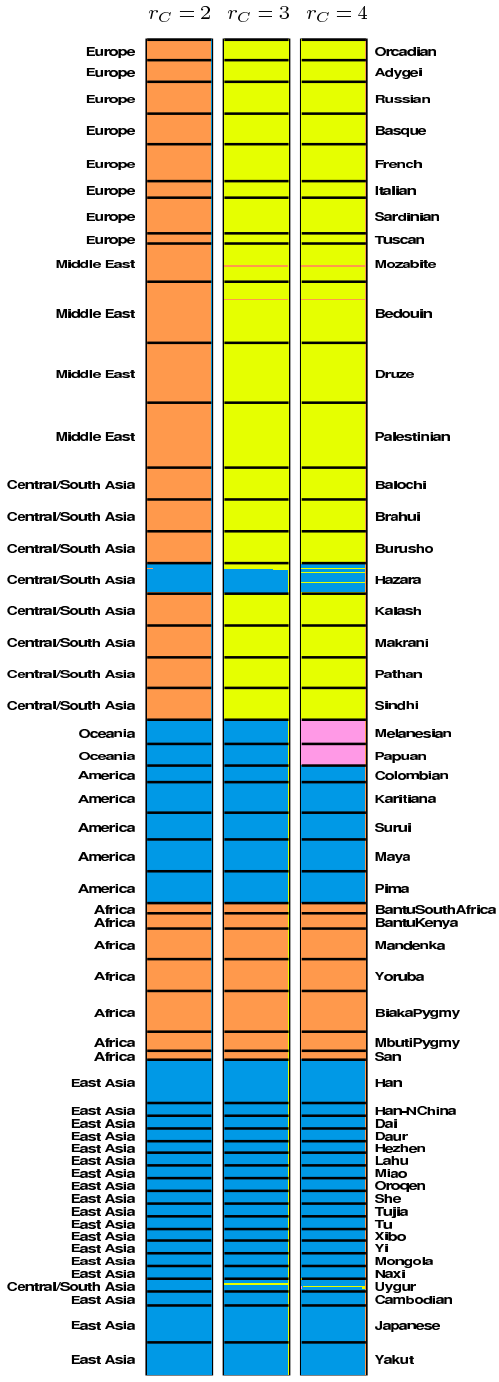


Fig. 8.2. Inferred population structure using the multi-start EMA.

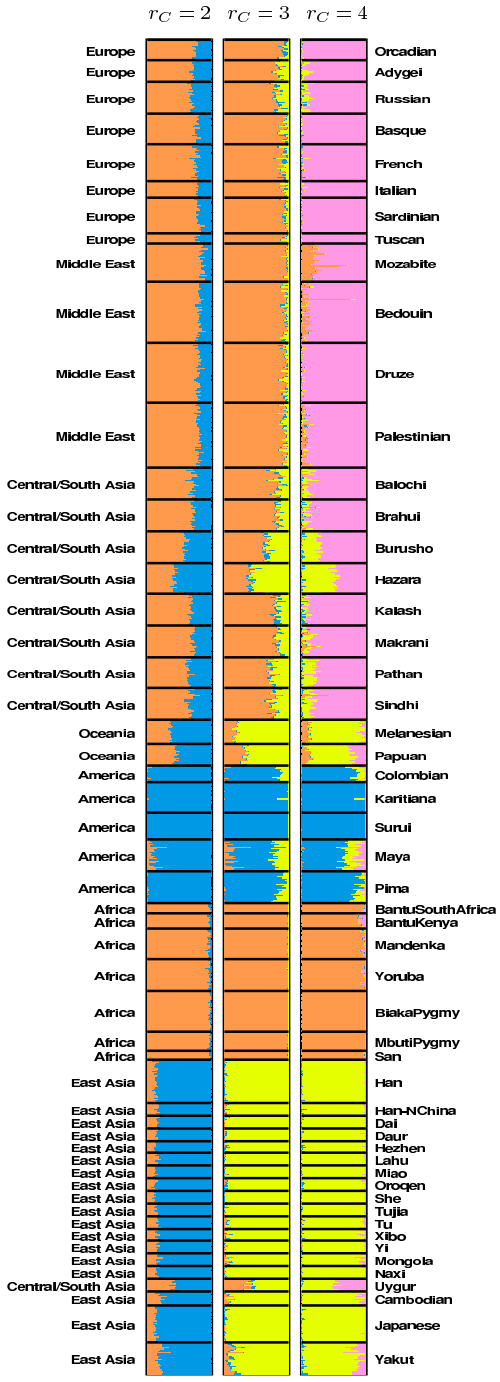


Fig. 8.3. Inferred population structure using STRUCTURE.

represents the estimated membership of the individual to each one of the r_C clusters. For each number of clusters, only the best run of ten on the basis of the F_{ST} measure is shown. In these experiments, the multi-start EMA tends to assign each individual to a cluster with a very high probability, being the membership probability for the rest of the clusters very small. Thus, the admixture proportions detected by multi-start EMA are also very small. By contrast, STRUCTURE detects a higher level of admixture among populations. The results obtained by STRUCTURE may be biologically correct since admixture is usual in population genetics. However, not always detected admixture proportions represents genuine contributions from corresponding ancestral sources since the uncertainty about the allele frequencies in two particular source populations may cause the overestimation of the admixture proportions (Corander and Marttinen, 2006). Moreover, although the admixed ancestry detected by multi-start EMA is very small, the genetic distance given by the F_{ST} metric[2] is much higher (see Tables 8.2 and 8.3). This suggests that the populations obtained by the multi-start EMA are genetically more distant between themselves than the populations obtained by STRUCTURE.

	F_{ST} values		
	$r_C = 2$	$r_C = 3$	$r_C = 4$
Run 1	0.02714400	0.03193830	0.03100300
Run 2	0.02714400	0.03193830	0.02937500
Run 3	0.02714400	0.03193830	0.03463900
Run 4	0.02714400	0.03193830	0.02989200
Run 5	0.02715100	0.03156800	0.03100300
Run 6	0.02714400	0.03193830	0.03100300
Run 7	0.02714400	0.03193830	0.03100300
Run 8	0.02714400	0.03541020	0.03100300
Run 9	0.02714400	0.03193830	0.03100300
Run 10	0.02715200	0.03193830	0.03905100
Mean	0.02714550	0.03224446	0.03189750
Std	0.00000317	0.00111700	0.00286500

Table 8.2. F_{ST} values obtained on ten runs of the multi-start EMA with two, three and four clusters ($r_C = 2$, $r_C = 3$ and $r_C = 4$). The mean and standard deviation values over the ten runs are also reported.

On the other hand, the population structure obtained by the multi-start EMA mainly correspond to major geographical regions. Nevertheless, it is surprising that most of the individuals from Uygur and Hazara ethnic group are classified in the same cluster as ethnic groups from East Asia even when they are located in Central/South Asia. Similarly, a few Mozabite individu-

[2] Since the dataset includes two different types of polymorphisms, F_{ST} genetic distance is calculated taking into account only genetic information regarding microsatellite markers and therefore, insertion/deletion polymorphisms are ignored.

	F_{ST} values		
	$r_C = 2$	$r_C = 3$	$r_C = 4$
Run 1	0.00069648	0.00315150	0.00579690
Run 2	0.00069517	0.00315350	0.00579590
Run 3	0.00069517	0.00392830	0.00579690
Run 4	0.00069517	0.00311270	0.00579690
Run 5	0.00069517	0.00315350	0.00579690
Run 6	0.00069517	0.00316680	0.00579690
Run 7	0.00069517	0.00313840	0.00579690
Run 8	0.00067972	0.00315350	0.00579690
Run 9	0.00069517	0.00315350	0.00582990
Run 10	0.00069517	0.00315350	0.00579690
Mean	0.00069375	0.00322652	0.00580010
Std	0.00000000	0.00024699	0.00001047

Table 8.3. F_{ST} values obtained on ten runs of STRUCTURE with two, three and four clusters ($r_C = 2$, $r_C = 3$ and $r_C = 4$). The mean and standard deviation values over the ten runs are also reported.

als are clustered into a group dominated by Africans. This situation may be apparently strange. However, these results obtained by the multi-start EMA agree with those obtained by STRUCTURE, where the membership coefficients of Hazara and Yaruba individuals are higher for the cluster dominated by East Asia individuals than for the cluster dominated by Central/South Asia individuals.

8.5.1 Number of Clusters in the Dataset

As it was stated before, the multi-start EMA and STRUCTURE assume that the number of clusters, r_C, is given. However, this is not usually true in real problems. In order to select the best number of clusters, we evaluate the results with two, three and four clusters. We have also extended the experiments to more clusters but the multi-start EMA algorithm, in these experiments, converges to different solutions in separate runs when the number of clusters is higher than four. Actually, a few runs with five clusters yield similar results than with four clusters but a new group is created with individuals of American origin (data not shown). These results are also similar to those obtained by Rosenberg et al. (2002,2005) with five clusters. Nevertheless, most of the runs with five and six clusters results in a partition with four clusters (similar to the one shown in Figure 8.2 with $r_C = 4$) and one or two empty clusters respectively. Therefore, the multi-start EMA is detecting that there is no more than four clear clusters.

In order to decide the number of clusters, we use the F_{ST} distance. Table 8.2 shows the F_{ST} values over ten independent runs of the multi-start EMA for each number of clusters (two, three and four). The best mean F_{ST} value corresponds to the experiments with three clusters. Clustering the data into

four groups produces, except for run 10, slightly lower F_{ST} values than with three clusters. However, the difference of the F_{ST} values with three and four clusters is not statistically very significant (the p-value of the Mann-Whitney test is 0.72). On the contrary, the difference between F_{ST} values with two and three clusters is statistically significant (p-value=0.00). Therefore, we think that according to the multi-start EMA and the proposed metric, there are three or four clear clusters underlying the dataset. These results are compatible with those presented in (Rosenberg *et al.*, 2005) where the quality of the clustering partition is given by the clusteredness[3]. Although Rosenberg et al. also consider the presence of more than four clusters, and the best partition, on the basis of the clusteredness, is obtained with only two clusters, the populations obtained with three and four clusters are considered as good partitions too.

8.5.2 Selection of Relevant Markers

The use of the multi-start EMA algorithm allows to select the most relevant markers needed to obtain the clustering partition. The number of selected markers is controlled by two parameters, p_{rel} and p_{red}, which represent the thresholds for the statistical tests, being p_{rel} the threshold to control the selection of relevant markers and p_{red} the threshold to control the redundancy between the selected markers. STRUCTURE software is not able to filter out irrelevant or redundant markers and the presence of this irrelevant and/or redundant information may damage its ability to obtain the clustering partition. In order to show how STRUCTURE software behaves in these situations and how the information about irrelevant and redundant markers provided by the multi-start EMA helps to obtain a better clustering partition, we proceed as follows: first, we select, for each number of clusters, the best model on the basis of the F_{ST} metric (these are the models used to obtain the population partitions represented in Figure 8.2). Then, we perform a selection of the most relevant markers by using the maker selection algorithm where the parameter of the test p_{rel} varies in $\{0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and p_{red} in $\{0, 0.01, 0.05, 0.1\}$. Each parameter configuration gives rise to a different set of selected markers (Figure 8.4). Finally, for each set of selected markers we run ten independent executions of STRUCTURE and measure the mean value over the ten runs of the F_{ST} metric.

Figure 8.5 shows the mean F_{ST} value for each one of the selected marker sets. A clear trend can be observed in the plots: marker selection using small values of p_{rel} and p_{red} improves the F_{ST} values obtained by STRUCTURE. Additionally, as the value of p_{rel} increases, the number of selected markers also increases including more redundant information. Consequently, the mean value of the F_{ST} metric decreases approaching the F_{ST} value obtained by

[3] Clustering quality metric that measures the extent to which individuals were estimated to belong to a single cluster rather than to a combination of clusters.

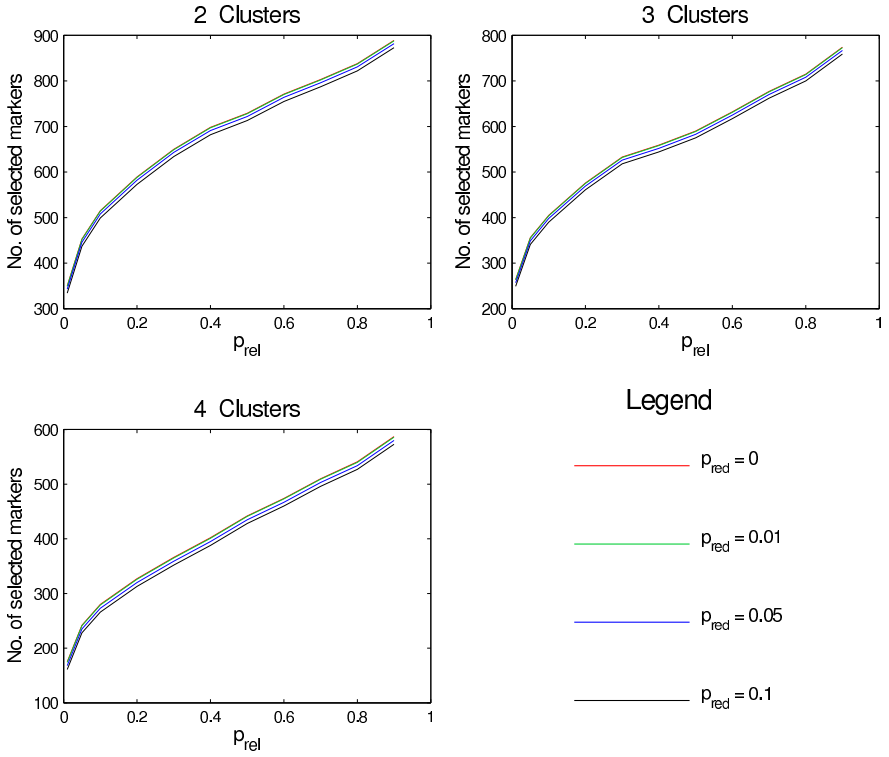


Figure 3. Number of Selected Markers.

Fig. 8.4. Number of selected markers as a function of p_{rel} and p_{red}.

STRUCTURE with the whole set of markers. It should be noted that the sets of selected markers are used only to obtain the clustering partition with STRUCTURE, but the F_{ST} metric is always calculated taking into account all the microsatellite markers.

According to the experimental results, we can say that the information about relevant markers implicitly included in the model calculated by the multi-start EMA helps to obtain the clustering partition. STRUCTURE, as well as other algorithms for the inference of population structure, does not take into account the existence of irrelevant and redundant information in the dataset. Therefore, the presence of irrelevant and/or redundant information damages their ability to retrieve the population structure underlying the dataset.

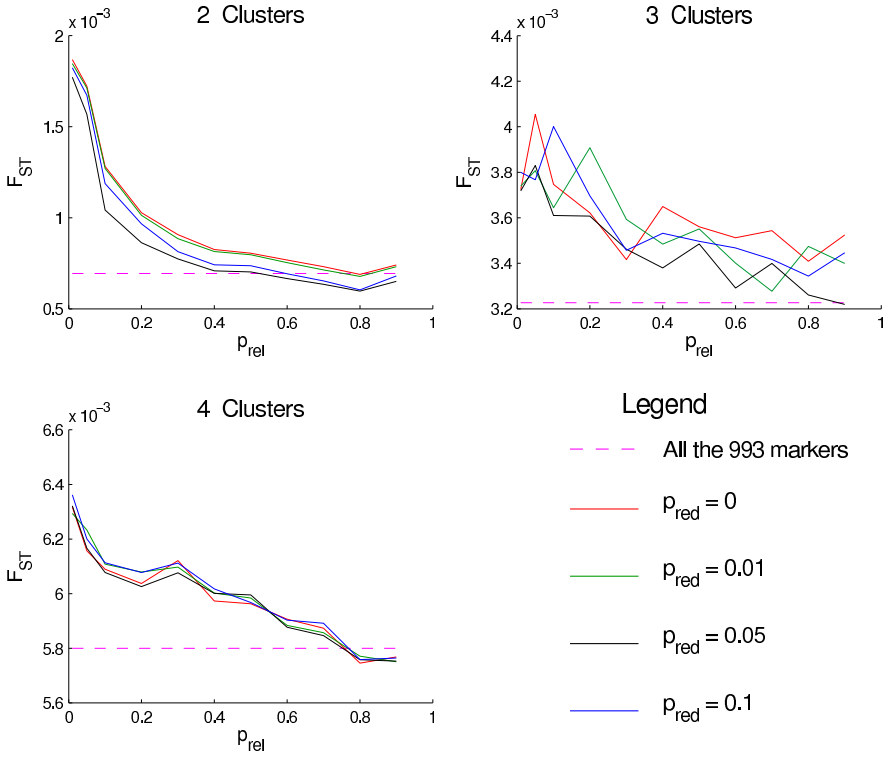


Fig. 8.5. Evolution of F_{ST} metric in the marker selection process.

8.6 Conclusions and Future Work

The inference of population structure is a very important and highly studied problem in population genetics. In this chapter we have described a new approach to infer the structure of a population and assign individuals (probabilistically) to populations. In order to approach the problem, we have tailored the EMA algorithm (introduced in Section 7.3.1) to work with polyploid and missing data. Thus, we provide a useful and realistic tool for the inference of population structure.

One of the advantages of the EMA algorithm with respect to other clustering techniques is the capability of dealing with irrelevant data for clustering purposes. This irrelevant information may damage the ability of other methods to obtain the underlying population structure. Moreover, we propose a marker selection algorithm based on mutual information which is able to obtain the most relevant markers needed to retrieve the population structure.

The performance of the multi-start EMA is evaluated in real problems and compared with STRUCTURE, which is the most widely used software for the inference of population structure. The results from the experiment with SNPs

data show that the EMA algorithm outperforms STRUCTURE in terms of accuracy. Additionally, the results from the experiment with microsatellites and insertion/deletion polymorphisms show that the populations obtained by the multi-start EMA have higher values for the F_{ST} metric than populations obtained by STRUCTURE. This suggests that populations obtained by the multi-start EMA are genetically more distant than populations obtained by STRUCTURE. By contrast, in this experiment, the multi-start EMA is not able to obtain population partitions with more than four clusters. It may suggest that the multi-start EMA does not perform as well as STRUCTURE when subpopulations of individuals in the dataset are genetically very close. Nevertheless, the EMA algorithm provides useful information about the relevant markers to infer the population structure. This information has been shown to be useful to improve the behavior of other methods such as STRUCTURE. Nevertheless, the selection of an informative subset of genetic markers that contains enough information to differentiate between populations under study may be very useful not only to correct population stratification in association studies or in admixture-mapping studies (Patterson *et al.*, 2004) but also to reduce the economical cost of sequencing samples for these studies.

One of the main drawbacks of the multi-start EMA, as well as other popular algorithms for the inference of population structure, is the fact that the number of clusters has to be fixed in advance. In order to overcome this multi-start EMA restriction, we provide a method to investigate the number of groups underlying the data. However, as was pointed out in Chapter 7, it may be very interesting to investigate other methods to automatically detect the number of clusters in the dataset.

On the other hand, the marker selection test proposed in this chapter only takes into account relationships between variables in pairs. It may also be interesting to investigate other methods to take into account more complex relationships among variables. Finally, the inference of population structure is not the only problem where the EMA algorithm can be applied. The special characteristics of the EMA as a clustering method make the algorithm an interesting tool to be applied to other bioinformatics problems such as the analysis of gene expression data or the analysis of mutations in genes related to specific diseases.

Part V

Conclusions

9

Conclusions

This chapter presents the general conclusions of this dissertation. More specific conclusions have been exposed at each corresponding chapter. Additionally, we include a list of publications and some future work.

Throughout the dissertation, we have focused the research on supervised classification and data clustering which are two fundamental fields of machine learning.

The part of the dissertation related to supervised classification is devoted to the discriminative learning of Bayesian network classifiers. We have shown that, usually, the discriminative learning is a better choice to learn classifiers because the learning process maximizes the conditional log-likelihood which is more related to the accuracy than the joint log-likelihood. However, the generative approach is much more efficient than the discriminative one and due to this efficiency, generative approach is sometimes preferred, especially when the model that we learn from data has the same structural restrictions as the model that generated the data. Nonetheless, the model that generated the data is usually unknown, therefore the discriminative approach may be more appropriate. In this part of the dissertation we have also introduced new algorithms to learn the parameters and structure of Bayesian network classifiers from a discriminative approach. We present the adaptation of the TM algorithm for the discriminative learning of the parameters as well as the construct-discriminative-TAN and structural TM for the discriminative learning of the structures.

On the other hand, the part related to data clustering is devoted to Bayesian model averaging. Bayesian model averaging is a technique that allows to deal with uncertainty in model selection. The presence of the hidden cluster variable in clustering problems makes the Bayesian model averaging calculations computationally intractable and only approximations are feasible. In fact, the approximations to Bayesian model averaging for clustering problems are usually given by approximations to the marginal likelihood obtained by means of Monte Carlo simulation or Laplace methods. In this dissertation we propose a new method to approximate the Bayesian model averaging over

restricted families of models such as naive Bayes and TAN models. These calculations can be obtained efficiently by means of the proposed algorithms: EMA and EMA-TAN. The empirical evaluation of the EMA algorithm shows that it is a powerful tool to learn Bayesian network models for clustering in situations where there is high uncertainty in model selection. This situation of uncertainty is crucial, for instance, in clustering problems where the dataset contains a lot of variables and only a few data samples. Another advantage of the EMA algorithm is that it includes in the learned model implicit information about which variables are more relevant for clustering purposes. Therefore, this information can be used to perform an unsupervised variable selection. Additionally, we present an application of the EMA algorithm to the inference of population structure. This is a problem taken from population genetics where human DNA polymorphisms are used to discover the DNA group structure in a pool of human individuals belonging to different ethnic groups distributed around the world. The experimental results show that the EMA algorithm can outperform other widely used methods to infer the structure of a population. Moreover, the information about the relevant variables for clustering purposes are useful to improve the results given by other algorithms.

9.1 List of Publications

The work presented in this dissertation has produced the following publications:

- G. Santafé J. A. Lozano, P. Larrañaga. Inference of population structure using genetic markers and a Bayesian model averaging approach for clustering. *Submitted to an international journal.*
- G. Santafé J. A. Lozano, P. Larrañaga. Discriminative vs. generative learning of Bayesian network classifiers. *Ninth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'2007)*, Hammamet, Tunisia, 2007. 453-464.
- G. Santafé J. A. Lozano, P. Larrañaga. Bayesian model averaging of naive Bayes for clustering. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, **36**(5), 1149-1161, 2006.
- G. Santafé J. A. Lozano, P. Larrañaga. Aprendizaje discriminativo de clasificadores Bayesianos. *Revista Iberoamericana de Inteligencia Artificial*, **29**(10), 39-47, 2006.
- P. Larrañaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armañanzas, G. Santafé A. Pérez, V. Robles. Machine learning in bioinformatics. *Briefings in Bioinformatics*, **7**(1), 86-112, 2006.
- I. Inza, R. Armañanzas, G. Santafé Una aproximación al software WEKA. *Aprendizaje Automático: Conceptos Básicos y Avanzados.* Editor B. Sierra, capítulo 23, 477-483. Pearson Educación, Madrid, 2006.

- G. Santafé J. A. Lozano, P. Larrañaga. Bayesian model averaging of TAN models for clustering. *European Workshop on Probabilistic Graphical Models (PGM 2006)*. Praha, Czech Republic, 2006. 271-278.
- G. Santafé J. A. Lozano, P. Larrañaga. Population substructure determination by means of Bayesian model averaging for clustering. *International Workshop on Intelligent Data Analysis in bioMedicine And Pharmacology (IDAMAP 2006)*. Verona, Italy, 2006. 51-56.
- G. Santafé J. A. Lozano, P. Larrañaga. Discriminative learning of Bayesian network classifiers via the TM algorithm. *Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005)*. Barcelona, Spain, 2005. 148-160.
- G. Santafé J. A. Lozano, P. Larrañaga. Aprendizaje discriminativo de clasificadores Bayesianos. *III Taller Nacional de Minería de Datos y Aprendizaje (TAMIDA 2005)*, Granada, Spain, 2005. 115-123.
- G. Santafé J. A. Lozano, P. Larrañaga. Full Bayesian Model Averaging of Naive Bayes for Clustering. Technical Report. EHU-KZAA-IK-3/04. University of the Basque Country, 2004.
- G. Santafé J. A. Lozano, P. Larrañaga. El Algoritmo TM para Clasificadores Bayesianos. Technical Report. EHU-KZAA-IK-2/04. University of the Basque Country, 2004.

9.2 Future Work

Undoubtedly, much research is to be undertaken in the areas of discriminative learning of Bayesian network classifiers and Bayes model averaging for clustering. The future work has been previously presented with the conclusions at the end of each corresponding chapter, but in this section we want to summarize the main open lines of research.

The interest for future developments related to the discriminative learning of Bayesian networks can be focused on the learning of structures. The performance of methods proposed in this dissertation is not as good as expected. The structural TM algorithm uses the conditional BIC score which is known to be quite restrictive and tends to select simple models. Similarly, the discriminative-construct-TAN algorithm maintains a TAN structure which can lead to include arcs that decrease the conditional log-likelihood score in the model. Moreover, both algorithms are greedy approaches to the discriminative learning of structures. It is interesting to investigate other metrics to guide the searching process of the structural TM as well as the use of a heuristic search to avoid the local maxima in the greedy search performed by both structural TM and construct-discriminative-TAN. Additionally, in order to allow the evaluation of complex models, the TM algorithm to learn the parameters should be extended to more complex structures and also can be extended to deal with missing values. Another very interesting future research could include the study, from a theoretical approach, of the limitations

of generative and discriminative learning to solve specific problems in order to decide which approach is better for a given problem.

Future work related to the Bayesian model averaging for clustering could include the extension of the expectation model averaging to other more complex models, the search for a good node ordering for the averaging over structures and the averaging over several node orders. Moreover, the EMA and EMA-TAN algorithms are limited by the fact that the number of clusters must be given in advance. Future modifications of the algorithms may avoid this limitation. Additionally, the convergence properties of the EMA algorithm has to be studied from a theoretical point of view. On the other hand, the EMA algorithm provides a powerful tool for unsupervised feature selection. In this dissertation we present an algorithm that uses the model learned by the EMA algorithm in order to select the most relevant variables for the clustering process. However, other methods to obtain information about the relevant variables from the model learned by the EMA algorithm can be developed.

Finally, in this dissertation, we present an application of the EMA algorithm to a real problem taken from population genetics, but there are many other real problems where the EMA algorithm can be applied such as the clustering of gene expression data.

References

Acid, S. and de Campos, L. M. (1995). *Advances in Intelligence Computing*, volume 945 of *Lecture Notes in Computer Science*, chapter Approximations of causal networks by polytrees: An empirical study, pages 149–158. Springer Verlag.

Acid, S., de Campos, L. M., and Castellano, J. G. (2005). Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs. *Machine Learning*, **59**, 213–235.

Aitkin, M., Anderson, D., and Hinde, J. (1981). Statistical modeling of data on teaching styles. *Journal of the Royal Statistical Society B*, **144**, 419–461.

Akaike, H. (1974). New look at the statistical model identification. *IEEE Transactions on Automatic Control*, **19**(6), 716–723.

Andenberg, M. R. (1973). *Clustering Analysis for Applications*. Academic Press.

Bamshad, M. J., Wooding, S., Watkins, W. S., Ostler, C. T., Batzer, M. A., and Jorde, L. B. (2003). Human population genetic structure and inference of group membership. *American Journal of Human Genetics*, **72**, 578–589.

Bandyopadhyay, S. (2005). Automatic determination of the number of fuzzy clusters using simulated annealing with variable representation. In *Proceedings of the Fifteenth International Symposium on Methodologies for Intelligent Systems*, pages 594–602.

Banfield, J. D. and Raftery, A. E. (1993). Model-based Gaussian and non-Gaussian clustering. *Biometrics*, **49**, 803–821.

Barash, Y. and Friedman, N. (2002). Context-specific Bayesian clustering for gene expression data. *Journal of Computational Biology*, **9**, 169–191.

Ben-Bassat, M. (1982). *Handbook of Statistics*, volume 2, chapter Use of distance measures, information measures and error bounds in feature evaluation, pages 773–791. North-Holland Publishing Company.

Bernardo, J. M. and Smith, A. F. M. (2000). *Bayesian Theory*. Wiley.

Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press.

Binder, D. A. (1978). Bayesian cluster analysis. *Biometrika*, **65**, 31–38.

Binder, D. A. (1981). Approximations to Bayesian clustering rules. *Biometrika*, **68**, 275–285.

Bishop, C. (1996). *Neural Networks for Pattern Recognition*. Oxford Press.

Blake, C. and Merz, C. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn.

Blanco, R. (2005). *Learning Bayesian Networks from Data with Factorisation and Classification Purposes. Applications in Biomedicine*. Ph.D. thesis, University of the Basque Country.

Blanco, R., Inza, I., and Larrañaga, P. (2003). Learning Bayesian networks in the space of structures by estimation of distribution algorithms. *International Journal of Intelligent Systems*, **18**, 205–220.

Blanco, R., Larrañaga, P., Inza, I., and Sierra, B. (2004). Gene selection for cancer classification using wrapper approaches. *International Journal of Pattern Recognition and Artificial Intelligence*, **18**(8), 1373–1390.

Blanco, R., Inza, I., Merino, M., Quiroga, J., and Larrañaga, P. (2005). Feature selection in Bayesian classifiers for the prognosis of survival of cirrhotic patients treated with TIPS. *Journal of Biomedical Informatics*, **9**(4), 406–423.

Bouckaert, R. (1994). A stratified simulation scheme for inference in Bayesian belief networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 110–117.

Bouckaert, R. R. (1995). *Bayesian Belief Networks: From Construction to Inference*. Ph.D. thesis, University of Utrecht.

Bowcock, A. M., Ruiz-Linares, A. A., Tomfohrde, J., Minch, E., Kidd, J., and Cavalli-Sforza, L. (1994). High resolution of human evolutionary trees with polymorphic microsatellites. *Nature*, **368**, 455–457.

Braga-Neto, U. and Dougherty, E. (2004). Bolstered error estimation. *Pattern Recognition*, **37**, 1267–1281.

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

Buntine, W. (1991). Theory refinement in Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60.

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

Cann, H. M., de Toma, C., Cazes, L., Legrand, M.-F., Morel, V., Piouffre, L., Bodmer, J., Bodmer, W. F., Bonne-Tamir, B., Cambon-Thomsen, A., Chen, Z., Chu, J., Carcassi, C., Contu, L., Du, R., Excoffier, L., Ferrara, G. B., Friedlaender, J. S., Groot, H., Gurwitz, D., Jenkins, T., Herrera, R. J., Huang, X., Kidd, J., Kidd, K. K., Langaney, A., Lin, A. A., Mehdi, S. Q., Parham, P., Piazza, A., Pistillo, M. P., Qian, Y., Shu, Q., Xu, J., Zhu, S., Weber, J. L., Greely, H. T., Feldman, M. W., Thomas, G., Dausset, J., and Cavalli-Sforza, L. L. (2002). A human genome diversity cell line panel. *Science*, **296**, 261–262.

Cardon, L. R. and Palmer, L. J. (2003). Population stratification and spurious allelic association. *The Lancet*, **361**, 598–604.

Castillo, E., Gutierrez, J. M., and Hadi, A. S. (1997). *Expert Systems and Probabilistic Network Models*. Springer-Verlag.

Cerquides, J. and López de Mántaras, R. (2003a). The indifferent naive Bayes classifier. In *Proceedings of the Sixteenth International FLAIRS Conference*, pages 341–345.

Cerquides, J. and López de Mántaras, R. (2003b). Tractable Bayesian learning of tree augmented naive Bayes models. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 75–82.

Cerquides, J. and López de Mántaras, R. (2005). TAN classifiers based on decomposable distributions. *Machine Learning*, **59**(3), 323–354.

Cestnik, B., Kononenko, B., and Bratko, I. (1987). *Progress in Machine Learning*, chapter ASSISTANT-86: A knowledge elicitation tool for sophisticated users, pages 31–45. Sigma Press.

Chatfield, C. (1995). Model uncertainty, data mining, and statistical inference (with discussion). *Journal of the Royal Statistical Society. Series A*, **158**, 419–466.

Chavez, R. and Cooper, G. (1990). A randomized approximation algorithm for probabilistic inference on Bayesian belief networks. *Networks*, **20**(5), 661–685.

Cheeseman, P. and Stutz, J. (1996). Bayesian classification (Autoclass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press.

Cheng, J. and Greiner, R. (2001). Learning Bayesian belief network classifiers: Algorithms and system. In *Proceedings of the Fourteenth Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, pages 141–152.

Chickering, D. M. and Heckerman, D. (1997). Efficient approximation for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, **29**, 181–212.

Chickering, D. M., Geiger, D., and Heckerman, D. (1994). Learning Bayesian networks is NP-hard. Technical Report MSR-TR-94-17, Microsoft Research.

Chickering, D. M., Geiger, D., and Heckerman, D. (1995). Learning Bayesian networks: Search methods and experimental results. In *Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 112–128.

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

Climescu-Haulica, A. (2006). *Advances in Data Analysis. Proceedings of the Thirtieth Annual Conference of the Gesellschaft für Klassifikation*, chapter How to choose the number of clusters: The Cramer multiplicity solution, pages 15–22. Springer.

Cooper, G. (1990). Computational complexity of probabilistic inference using Bayesian belief networks (Research note). *Artificial Intelligence*, **42**, 393–405.

Cooper, G. F. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, **9**, 309–347.

Corander, J. and Marttinen, P. (2006). Bayesian identification of admixture events using multilocus molecular markers. *Molecular Ecology*, **15**, 2833–2843.

Corander, J., Waldmann, P., Marttinen, P., and Sillanpää, M. J. (2004). BAPS 2: Enhanced possibilities for the analysis of genetic population structure. *Bioinformatics*, **20**(15), 2363–2369.

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*. John Wiley & Sons, second edition.

Dagum, P. and Luby, M. (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, **60**, 141–153.

Dasgupta, A. and Raftery, A. E. (1998). Detecting features in spatial point processes with clutter via model-based clustering. *Journal of American Statistical Association*, **93**(441), 294–302.

Dash, D. and Cooper, G. (2002). Exact model averaging with naive Bayesian classifiers. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 91–98.

Dash, D. and Cooper, G. (2003). Model averaging with discrete Bayesian network classifiers. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*.

Dash, D. and Cooper, G. F. (2004). Model averaging for prediction with discrete Bayesian networks. *Journal of Machine Learning Research*, **5**, 1177–1203.

Dawid, A. P. (1976). Properties of diagnostic data distributions. *Biometrics*, **32**, 647–658.

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

Dawson, K. J. and Belkhir, K. (2001). A Bayesian approach to the identification of panmictic populations and the assignment of individuals. *Genetical Research*, **78**, 59–77.

de Campos, L. M. (1998). *Probabilistic Expert Systems*, chapter Automatic learning of graphical models I: Basic methods, pages 113–140. Ediciones de la Universidad de Castilla-La Mancha. In Spanish.

de Campos, L. M. and Puerta, J. M. (2001). Stochastic local and distributed search algorithms for learning Bayesian networks. In *Proceedings of the Third Symposium on Adaptive Systems*, pages 109–115.

de Campos, L. M., Fernández-Luna, J. M., Gámez, J. A., and Puerta, J. M. (2002). Ant colony optimization for learning Bayesian networks. *International Journal of Approximate Reasoning*, **31**(21), 291–311.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, **39**, 1–38.

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

Dietterich, T. G. (1998). Approximate statistical test for comparing supervised learning algorithms. *Neural Computation*, **18**, 1895–1924.

Díez, F. J. (1996). Local conditioning in Bayesian networks. *Artificial Intelligence*, **87**, 1–20.

Dijkstra, T. K. (1988). *On Model Uncertainty and its Statistical Implications*. Springer.

Doak, J. (1992). An evaluation of feature selection methods and their application to computer security. Technical Report CSE-92-18, University of California at Davis.

Domingos, P. (2000). Bayesian averaging of classifiers and the overfitting problem. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 223–230.

Domingos, P. and Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, **29**(2-3), 103–130.

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis*. John Wiley and Sons.

Edwards, D. and Lauritzen, S. L. (2001). The TM algorithm for maximising a conditional likelihood function. *Biometrika*, **88**(4), 961–972.

Efron, B. and Tibshirani, R. (1993). *An Introduction to Bootstrap*. Chapman and Hall.

Elidan, G. and Friedman, N. (2001). Learning the dimensionality of hidden variables. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 144–151.

Etxeberria, R., Larrañaga, P., and Pikaza, J. M. (1998). Analysis of the behavior of the genetic algorithms when searching Bayesian networks from data. *Pattern Recognition Letters*, **18**(11-13), 1269–1273.

Everitt, B. S. (1980). *Cluster Analysis*. Heineman Educational Books.

Ezawa, K., Singh, M., and Norton, S. (1996). Learning goal oriented Bayesian networks for telecommunications risk management. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 139–147.

Falush, D., Stephens, M., and Pritchard, J. K. (2003). Inference of population structure using multilocus genotype data: Linked loci and correlated allele frequencies. *Genetics*, **164**, 1567–1587.

Fayyad, U. and Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1027.

Feelders, A. and Ivanovs, J. (2006). Discriminative scoring of Bayesian network classifiers: A comparative study. In *Proceedings of the Third European Workshop on Probabilistic Graphical Models*, pages 75–82.

Forgy, E. (1965). Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, **21**, 768–769.

Fraley, C. and Raftery, A. E. (1998). How many clusters? Which clustering methods? Answers via model-based clustering analysis. *Computer Journal*, **41**, 578–588.

Friedman, N. (1997). Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 125–133.

Friedman, N. (1998). The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 129–138.

Friedman, N. and Koller, D. (2003). Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, **50**, 95–126.

Friedman, N. and Pe'er, M. L. I. N. D. (2000). Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, **7**(3-4), 601–620.

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

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Academic Press.

Gammerman, A. and Thatcher, A. (1991). Bayesian diagnostic probabilities without assuming independence of symptoms. *Methods of Information in Medicine*, **30**, 15–22.

Gangnon, R. and Clayton, M. (2007). Cluster detection using Bayes factors from overparameterized cluster models. *Environmental and Ecological Statistics*, **14**(1), 69–82.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

Geiger, D. (1992). An entropy-based learning algorithm of Bayesian conditional trees. In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 92–97.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 721–742.

Gilks, W., Richardson, S., and Spiegelhalter, editors (1998). *Markov Chain Monte Carlo in Practice*. CRC Press.

Giudici, P. and Green, P. J. (1999). Decomposable graphical Gaussian model determination. *Biometrika*, **86**(4), 785–801.

Golub, T. R., Slonim, D. K., Tamayo, P., Juard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomgield, C. D., and Lander, E. S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531–537.

Greiner, R., Zhou, W., Su, X., and Shen, B. (2005). Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, **59**(3), 297–322.

Grossman, D. and Domingos, P. (2004). Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of the Twenty First International Conference on Machine Learning*, pages 361–368.

Guillot, G., Estoup, A., Mortier, F., and Cosson, J. (2004). A spatial statistical model for landscape genetics. *Genetics*, **170**, 1261–1280.

Guo, Y. and Greiner, R. (2005). Discriminative model selection for belief net structures. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 296–305.

Hamerly, G. and Elkan, C. (2001). Bayesian approaches of failure prediction for disk drives. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 202–209.

Hand, D. and You, K. (2001). Idiot's Bayes -not so stupid after all? *International Statistical Review*, **69**, 385–398.

Hartigan, J. A. (1975). *Clustering Algorithms*. John Wiley and Sons.

Hartley, H. (1958). Maximum likelihood estimation from incomplete data. *Biometrics*, **14**, 174–194.

Hastie, T. and Tibshirani, R. (1990). *Generalized Additive Models*. Chapman Hall.

Heckerman, D. (1995). A tutorial on learning with Bayesian networks. Technical report, Microsoft Research. MSR-TR-95-06.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, **20**, 197–243.

Henrion, M. (1988). Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Proceedings of the Second Conference on Uncertainty in Artificial Intelligence*, pages 149–163.

Herskovits, E. and Cooper, G. (1990). Kutató: An entropy-driven system for construction of probabilistic expert systems from database. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 54–62.

Hinds, D. A., Stuve, L. L., Nilsen, G. B., Halperin, E., Eskin, E., Ballinger, D. G., Frazer, K. A., and Cox, D. R. (2005). Whole-genome patterns of common DNA variation in three human populations. *Science*, **307**, 1072–1079.

Hoeting, J., Madigan, D., Raftery, A. E., and Volinsky, C. (1999). Bayesian model averaging: A tutorial. *Statistical Science*, **14**, 382–417.

Hosmer, D. and Lemeshow, S. (1989). *Applied Logistic Regression*. John Wiley and Sons.

Hryceij, T. (1990). Gibbs sampling in Bayesian networks. *Artificial Intelligence*, **46**(3), 351–363.

Huang, K., King, I., and Lyu, M. R. (2003). Discriminative training of Bayesian Chow-Liu tree multinet classifiers. In *Proceedings of the International Joint Conference on Neural Network*, pages 484–488.

Huang, K., Zhou, Z., King, I., and Lyu, M. R. (2005). Improving naive Bayesian classifier by discriminative training. In *Proceedings of the Twelfth International Conference on Neural Information Processing*, pages 49–54.

Hwang, K. and Zhang, B. (2005). Bayesian model averaging of Bayesian networks classifiers over multiple node-orders: Applications to sparse datasets. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, **35**(6), 1302–1310.

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

Inza, I., Larrañaga, P., Etxeberria, R., and Sierra, B. (2000). Feature subset selection by Bayesian networks based optimization. *Artificial Intelligence*, **123**(1-2), 157–184.

Inza, I., Larrañaga, P., and Sierra, B. (2001a). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, chapter Feature subset selection by estimation of distribution algorithms, pages 265–290. Kluwer Academic Publishers.

Inza, I., Larrañaga, P., and Sierra, B. (2001b). Feature subset selection by Bayesian networks: a comparison with genetic and sequential algorithms. *International Journal of Approximate Reasoning*, **27**(2), 143–164.

Inza, I., Merino, M., Larrañaga, P., Quiroga, J., Sierra, B., and Girala, M. (2001c). Feature subset selection by genetic algorithms and estimation of distribution algorithms. A case study in the survival of cirrhotic patients treated with TIPS. *Artificial Intelligence in Medicine*, **23**(2), 187–205.

Inza, I., Sierra, B., Blanco, R., and Larrañaga, P. (2002). Gene selection by sequential wrapper approaches in microarray cancer class prediction. *Journal of Intelligent and Fuzzy Systems*, **12**(1), 25–34.

Inza, I., Larrañaga, P., Blanco, R., and Cerrolaza, A. J. (2004). Filter versus wrapper gene selection approaches in DNA microarray domains. *Artificial Intelligence in Medicine*, **31**(2), 91–103.

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice-Hall.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, **31**(3), 264–323.

Jebara, T. (2003). *Machine Learning: Discriminative and Generative*. Kluwer Academic Publishers.

Jensen, F. (2001). *Bayesian Networks and Decision Graphs*. Springer Verlag.

Jensen, F. and Andersen, S. (1990). Approximations in Bayesian belief universes for knowledge based systems. Technical report, Institute of Electronic Systems, Aalborg University.

Jensen, F., Olesen, K., and Andersen, S. (1990a). An algebra of Bayesian belief universes for knowledge based systems. *Networks*, **20**, 637–659.

Jensen, F., Lauritzen, S., and Olesen, K. (1990b). Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, **5**(4), 2269–282.

Jensen, F. V. and Nielsen, T. D. (2007). *Bayesian Networks and Decision Graphs*. Springer Verlag, second edition.

Jordan, M. I., editor (1998). *Learning in Graphical Models*. Kluwer.

Karciauskas, G., Kocka, T., Jensen, F. V., Larrañaga, P., and Lozano, J. A. (2004). Learning of latent class models by splitting and merging components. In *Second Workshop on Probabilistic Graphical Models*.

Keogh, E. J. and Pazzani, M. (1999). Learning augmented Bayesian classifiers: A comparison of distribution-based and non distribution-based approaches. In *Proceedings of the Seventeenth International Workshop on Artificial Intelligence and Statistics*, pages 225–230.

Kohavi, R. (1995). *Wrapper for Performance Enhancement and Oblivious Decision Graphs*. Ph.D. thesis, Stanford University.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, **97**(1-2), 273–324.

Kohonen, T. (2001). *Self-Organizing Maps*. Springer-Verlag, third edition.

Kollin, J. and Koivisto, M. (2006). Bayesian learning with mixtures of trees. In *Proceedings of the Seventeenth European Conference on Machine Learning*, pages 294–305.

Kong, C. J. A. and Kjærulff, U. (1993). Blocking gibbs sampling in very large probabilistic expert systems. Technical Report R 13-2031, Department of Mathematics and Computer Science, University of Aalborg.

Kononenko, I. (1990). *Current Trends in Knowledge Acquisition*, chapter Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. IOS Press.

Korb, K. B. and Nicholson, A. E. (2004). *Bayesian Artificial Intelligence*. Chapman & Hall / CRC.

Kullback, S. and Leibler, R. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, **22**, 79–86.

Lachenbruch, P. and Michey, R. (1968). Estimation error rates in discriminant analysis. *Technometrics*, **10**, 1–11.

Lam, W. and Bacchus, F. (1994). Learning Bayesian belief networks. An approach based on the MDL principle. *Computational Intelligence*, **10**(4), 269–293.

Langley, P. and Sage, S. (1994). Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406.

Larrañaga, P. (2004). Clasificación supervisada via modelos gráficos probabilísticos. Research work for the full professor position. In Spanish.

Larrañaga, P. and Lozano, J. A., editors (2002). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers.

Larrañaga, P., Kuijpers, C., Murga, R., and Yurramendi, Y. (1996). Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on System, Man and Cybernetics*, **26**(4), 487–493.

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

Larrañaga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J. A., Armañanzas, R., Santafé, G., Pérez, A., and Robles, V. (2006). Machine learning in bioinformatics. *Briefings in Bioinformatics*, **7**(1), 86–112.

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

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

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

Leamer, E. E. (1978). *Specification Searches*. Wiley.

Long, P. M., Servedio, R. A., and Simon, H. U. (2006). Discriminative learning can succeed where generative learning fails. *Information Processing Letters*, **103**(4), 131–135.

Lucas, P. (2004). *Advances in Bayesian Networks*, chapter Restricted Bayesian network structure learning, pages 217–234. Springer-Verlag.

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability I*, pages 281–297.

Madigan, D. and Raftery, A. E. (1994). Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, **89**, 1535–1546.

Madigan, D. and York, J. (1995). Bayesian graphical model for discrete data. *International Statistical Reviews*, **63**, 215–232.

Madigan, D., Raftery, A. E., York, J. C., Bradshaw, J. M., and Almond, R. G. (1994). Strategies for graphical model selection. In *Selecting Models from Data: Artificial Intelligence and Statistics*, pages 91–100.

Mani, S., Pazzani, M., and West, J. (1997). Knowledge discovery from a breast cancer database. In *Proceedings of the Sixth Conference on Artificial Intelligence in Medicine in Europe*, volume 1211 of *Lecture Notes in Computer Science*, pages 130–133. Springer-Verlag.

Mann, H. and Whitney, D. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, **18**, 50–60.

McCallum, A. and Nigam, K. (1998). A comparison on event models for naive bayes text classification. In *Proceedings of the Fifteenth American*

Association for Artificial Intelligence. Workshop on Learning for Text Categorization, pages 41–48.

McCarthy, J. (1956). Measures of the value of information. In *Proceedings of the National Academy of Sciences*, pages 645–655.

McGill, W. J. (1954). Multivariate information transmission. *Psychometrika*, **19**(2), 97–116.

McLachlan, G. and Krishnan, T. (1997). *The EM Algorithm and Extensions*. Wiley, New York.

McLachlan, G. and Peel, D. (2000). *Finite Mixture Models*. John Wiley & Sons.

McLachlan, G. J. and Basford, K. E. (1989). *Mixture models: Inference and applications to clustering*. Marcel Dekker Inc.

Medvedovic, M. and Guo, J. (2005). Bayesian model-averaging in unsupervised learning from microarray data. In *Proceedings of the Fourth Workshop on Data Mining in Bioinformatics*, pages 40–47.

Meila, M. and Heckerman, D. (1998). An experimental comparison of several clustering and initialization methods. Technical Report MSR-TR-98-06, Microsoft Research.

Meila, M. and Jaakkola, T. (2000). Tractable Bayesian learning of tree belief networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 380–383.

Meila, M. and Jaakkola, T. (2006). Tractable Bayesian learning of tree belief networks. *Statistics and Computing*, **16**(1), 77–92.

Meila, M. and Jordan, M. I. (1998). Estimating dependency structure as a hidden variable. In *Neural Information Processing Systems*, volume 10, pages 584–590.

Metropolis, N. and Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association*, **44**, 335–341.

Milligan, G. W. (1980). An estimation of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika*, **45**, 325–242.

Milligan, G. W. and Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, **58**(2), 159–179.

Minka, T. (2002). Bayesian model averaging is not model combination. Technical report, MIT Media Lab note.

Minka, T. (2005). Discriminative models, not discriminative training. Technical Report TR-2005-144, Microsoft Research Cambridge.

Minsky, M. (1961). Steps toward artificial intelligence. *Transactions on Institute of Radio Engineers*, **49**, 8–30.

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

Miyahara, K. and Pazzani, M. (2000). Collaborative filtering with the simple Bayesian classifier. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, pages 679–689.

Movellan, J., Wachtler, T., Albright, T., and Sejnowski, T. (2002). Naive Bayes coding of color in primary visual cortex. In *Proceedings of the Neural Information Processing Systems*.

Mühlenbein, H. and Mahning, T. (2001). *Advances in Evolutionary Synthesis of Intelligent Agents*, chapter Evolutionary synthesis of Bayesian networks for optimization, pages 429–455. MIT Press.

Myers, J. W., Laskey, K. B., and Levitt, T. (1999). Learning Bayesian networks from incomplete data with stochastic search algorithms. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 476–485.

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

Narasimhan, M. and Bilmes, J. (2005). A submodular-supermodular procedure with applications to discriminative structure learning. In *Proceedings of the Twenty First Conference on Uncertainty in Artificial Intelligence*, pages 434–441.

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

Ng, A. (1997). Preventing overfitting of cross-validation data. In *Proceedings of Fourteenth International Conference on Machine Learning*, pages 245–253.

Ohmann, C., Yang, Q., Kunneke, M., Stozing, H., Tohn, K., and Lorenz, W. (1988). Bayes theorem and conditional dependence of symptoms: Different models applied to data of upper gastrointestinal bleeding. *Methods of Information in Medicine*, **8**, 23–36.

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

O'Rourke, S., Chechik, G., and Eskin, E. (2005). Separation of overlapping subpopulations by mutual information. In *Proceedings of the NIPS Workshop on Computational Biology and the Analysis of Heterogeneous Data*.

Pardo, L. (1997). *Teoría de la Información Estadística*. Hespérides.

Patterson, N., Hattangadi, N., Lane, B., Lohmuller, K. E., Hafler, D. A., Oksenberg, J. R., Hauser, S. L., Simith, M. W., O'Brien, S. J., Altshuler, D., Daly, J., and Reich, D. (2004). Methods for high-density admixture mapping of disease genes. *American Journal of Human Genetics*, **74**, 1001–1013.

Pazzani, M., Murumatsu, J., and Billsus, D. (1996). Syskill and webert: Identifying interesting web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 54–61.

Peña, J., Lozano, J., and Larrañaga, P. (2000). An improved Bayesian structural EM algorithm for learning Bayesian networks for clustering. *Pattern Recognition Letters*, **21**(8), 779–786.

Peña, J. M., Lozano, J. A., and Larrañaga, P. (1999a). An empirical comparison of four initialization methods for the K-means algorithm. *Pattern Recognition Letters*, **20**(10), 1027–1230.

Peña, J. M., Lozano, J. A., and Larraaga, P. (1999b). Learning Bayesian networks for clustering by means of constructive induction. *Pattern Recognition Letters*, **20**(11-13), 1219–1230.

Pearl, J. (1986a). A constraint-propagation approach to probabilistic reasoning. In *Proceedings of the Second Conference in Uncertainty in Artificial Intelligence*, pages 357–369.

Pearl, J. (1986b). Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, **29**, 241–288.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Perez, A., Larrañaga, P., and Inza, I. (2006). Information theory and classification error in probabilistic classifiers. In *Discovery Science 2006, Lecture Notes in Artificial Intelligence 4265*, pages 347–351.

Perez, A., Larrañaga, P., and Inza, I. (2007). Discriminative structural learning of tree-augmented naive Bayes classifier that maximizes the conditional likelihood. Technical report, University of the Basque Country.

Pernkopf, F. and Bilmes, J. (2005). Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 657–664.

Price1, A. L., Patterson, N. J., Plenge, R. M., Weinblatt, M. E., Shadick, N. A., and Reich, D. (2006). Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics*, **38**, 904 – 909.

Pritchard, J., Stephens, M., Rosenberg, N., and Donnelly, P. (2000a). Association mapping in structured populations. *American Journal of Human Genetics*, **67**, 170–181.

Pritchard, J. K., Stephens, M., and Donnelly, P. (2000b). Inference of population structure using multilocus genotype data. *Genetics*, **155**, 945–959.

Provost, F., Fawcett, T., and Kohavi, R. (1998). The cases against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453.

Raiffa, H. and Schalaifer, R. (1961). *Applied Statistical Decision Theory*. Harvard Business School Publications.

Rao, J. and Shao, A. (1992). Jackknife variance estimation with survey data under hot deck imputation. *Biometrika*, **79**, 811–822.

Reynolds, J., Weir, B. S., and Cockerham, C. C. (1983). Estimation of the coancestry coefficient: Basis for a short-term genetic distance. *Genetics*, **105**, 767–779.

Riddle, E. L., Murthy, K. K., Eskin, E., O'Connor, D. T., Rana, B. K., and Insel, P. A. (2006). Single nucleotide polymorphisms and ethnic-specific haplotypes of the regulator of G-protein signaling-2 (rgs2) gene in hypertensive and normotensive subjects. *Hypertension*, **47**(3), 415–420.

Ripley, B. (1996). *Pattern Recognition and Neural Networks*. Cambridge University.

Rissanen, J. (1978). Modeling by shortest data description. *Automatics*, **14**, 465–471.

Roos, T., Wettig, H., Grünwald, P., Myllymäki, P., and Tirri, H. (2005). On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, **59**(3), 267–296.

Rosenberg, N. A. (2004). Distruct: A program for the graphical display of population structure. *Molecular Ecology Notes*, **4**, 137–138.

Rosenberg, N. A. (2005). Algorithms for selecting informative marker panels for population assignment. *Journal of Computational Biology*, **12**(9), 1183–1201.

Rosenberg, N. A., Pritchard, J. K., Weber, J. L., Cann, H. M., Kidd, K. K., Zhivotovsky, L. A., and Feldman, M. W. (2002). Genetic structure of human population. *Science*, **298**, 2381–2385.

Rosenberg, N. A., Li, L. M., Ward, R., and Pritchard, J. K. (2003). Informativeness of genetic markers for inference of ancestry. *American Journal of Human Genetics*, **73**, 1402–1422.

Rosenberg, N. A., Mahajan, S., Ramachandran, S., Zhao, C., Pritchard, J. K., and Fedman, M. W. (2005). Clines, clusters, and the effect of study design on the inference of human population structure. *PLoS Genetics*, **1**, 661–671.

Rubinstein, Y. D. and Hastie, T. (1997). Discriminative vs. informative learning. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 49–53.

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

Salmerón, A., Cano, A., and Moral, S. (2000). Importance sampling in Bayesian networks using probability trees. *Computational Statistics and Data Analysis*, **34**, 387–413.

Santafé, G., Lozano, J. A., and Larrañaga, P. (2004). El algoritmo TM para clasificadores Bayesianos. Technical Report EHU-KZAA-IK-2/04, University of the Basque Country. in Spanish.

Santafé, G., Lozano, J. A., and Larrañaga, P. (2005a). Aprendizaje discriminativo de clasificadores Bayesianos. In *III Taller de Minería de Datos y Aprendizaje*, pages 115–124. In Spanish.

Santafé, G., Lozano, J. A., and Larrañaga, P. (2005b). Discriminative learning of Bayesian network classifiers via the TM algorithm. In *Proceedings of the Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 148–160.

Santafé, G., Lozano, J. A., and Larrañaga, P. (2006a). Aprendizaje discriminativo de clasificadores Bayesianos. *Revista Iberoamericana de Inteligencia Artificial*, **29**(10), 39–47. In Spanish.

Santafé, G., Lozano, J. A., and Larrañaga, P. (2006b). Bayesian model averaging of naive Bayes for clustering. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, **36**, 1149–1161.

Santafé, G., Lozano, J. A., and Larrañaga, P. (2006c). Bayesian model averaging of TAN models for clustering. In *Proceedings of the Third European Workshop on Probabilistic Graphical Models*, pages 271–278.

Santafé, G., Lozano, J. A., Larrañaga, P., and Eskin, E. (2006d). Population substructure determination by means of bayesian model averaging for clustering. In *Proceedings of the Eleventh Workshop on Intelligent Data Analysis in Biomedicine and Pharmacology*, pages 51–56.

Santafe, G., Lozano, J. A., and Larrañaga, P. (2007). Discriminative vs. generative learning of bayesian network classifiers. In *Proceedings of the Ninth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 453–464.

Santafé, G., Lozano, J. A., and Larrañaga, P. (2007). Inference of population structure using genetic markers and a bayesian model averaging approach for clustering. Submitted.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, **6**, 461–464.

Shachter, R. (1988). Probabilistic inference and influence diagrams. *Operations Research*, **36**, 589–604.

Shereider, Y. A. (1964). *Method of Statistical Testing: Monte Carlo Method*. Elsevier Horth-Holland.

Sillanpää, M. J., Kilpikari, R., Ripati, S., Onkamo, P., and Uimari, P. (2001). Bayesian association mapping for quantitative traits in a mixture of two populations. *Genetic Epidemiology*, **21**, S692–S699.

Sneath, P. H. and Sokal, R. R. (1973). *Numerical Taxonomy*. Freeman.

Sobol, I. M. (1984). *The Monte Carlo Method*. Mir Publishers, Moscow.

Spiegelhalter, D. and Lauritzen, S. (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks*, **20**, 579–605.

Spirtes, P., Glymour, C., and Scheines, R. (1993). *Causation, Prediction and Search*, volume 81 of *Lecture Notes in Statistics*. Springer-Verlag.

Spirtes, S., Glymour, C., and Scheines, R. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computing Reviews*, **9**, 62–72.

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

Sundberg, R. (2002). The convergence rate of the TM algorithm of Edwards and Lauritzen. *Biometrika*, **89**, 478–483.

Tanner, M. (1996). *Tool for Statistical Inference*. Springer Verlag.

Thiesson, B., Meek, C., Chickering, D. M., and Heckerman, D. (1998). Learning mixtures of DAG models. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 504–513.

Todd, B. and Stamper, R. (1994). The relative accuracy of a variety of medical diagnostic programs. *Methods of Information in Medicine*, **33**, 402–416.

Turakulov, R. and Easteal, S. (2003). Number of SNPs loci needed to detect population structure. *Human Heredity*, **55**, 37–45.

Vapnik, V. (1998). *Statistical Learning Theory*. John Wiley and Sons.

Vinciotti, V., Tucker, A., Kellam, P., and Liu, X. (2006). The robust selection of predictive genes via a simple classifier. *Applied Bioinformatics*, **5**(1), 1–11.

Vogl, C., Sanchez-Cabo, F., Stocker, G., Hubbard, S., Wolkenhauer, O., and Trajanoski, Z. (2005). A fully Bayesian model to cluster gene-expression profiles. *Bioinformatics*, **21 (Suplement 2)**, ii130–ii135.

Watanabe, S. (1960). Information theoretical analysis of multivariate correlation. *IBM Journal of Research and Development*, **4**, 66–82.

Wei, L. and Traore, I. (2005). Determining the optimal number of clusters using a new evolutionary algorithm. In *Proceedings of Seventeenth IEEE International Conference on Tools with Artificial Intelligence*, pages 14–16.

Weir, B. (1996). *Genetic Data Analysis II: Methods for Discrete Population Genetic Data*. Sinauer Associates, second edition.

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

Wilcoxon, F. (1945). Individual comparison by ranking methods. *Biometrics*, **1**, 80–83.

Wu, B., Liu, N., and Zhao, H. (2006). Psmix: An R package for population structure inference via maximum likelihood method. *BMC Bioinformatics*, **7**, 317.

Yeung, K. Y., Bumgarner, R. E., and Raftery, A. E. (2005). Bayesian model averaging: Development of an improved multiclass, gene selection and classification tool for microarray data. *Bioinformatics*, **21**, 2394–2402.

Yuille, A. L. and Rangarajan, A. (2002). *Advances in Neural Information Processing Systems*, chapter The Concave-Convex Procedure (CCCP). MIT Press.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Controls*, **8**, 338–353.

Zhang, H. and Ling, C. (2001). Learnability of augmented naive Bayes in nominal domains. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 617–623.